RL-TR-96-199
Final Technical Report
December 1996

IMPACT OF CHANNEL PROPAGATION ON IMAGE COMPRESSION (ICPIC) A GRAPHICAL SIMULATION TOOL

The Analytic Sciences Corporation

Charles Drutman and John Delmedico

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-96-199 has been reviewed and is approved for publication.

APPROVED: Richard N. Smith
RICHARD N. SMITH
Project Engineer

FOR THE COMMANDER: John A. Graniero
JOHN A. GRANIERO
Chief Scientist
Command, Control, & Communications Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	Final Jun 95 - Jun 96

4. TITLE AND SUBTITLE

IMPACT OF CHANNEL PROPAGATION ON IMAGE COMPRESSION (ICPIC) A GRAPHICAL SIMULATION TOOL

5. FUNDING NUMBERS

C - F30602-95-C-0007
PE - N/A
PR - R450
TA - 01
WU - 08

6. AUTHOR(S)

TASC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TASC
555 French Road
New Hartford, NY 13413-0895

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/C3BA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-96-199

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Richard N. Smith/C3BA/(315) 330-7436

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Using Signal Processing Worksystem (SPW), a Computer Aided Design (CAD) tool, the performance of a wavelet compression algorithm was evaluated over wireless links. Visual and Least Mean Square (LMS) techniques were used to evaluate the performance of the compression algorithms.

14. SUBJECT TERMS

Image compression, Transmission of compressed imagery, wireless communications

15. NUMBER OF PAGES 220

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

1.0 Introduction

The ICPIC workstation is an extensible simulation testbed, with a graphical, menu-driven interface, that facilitates plug and play evaluation and design of optimal image compression and communication system configurations. The ICPIC testbed can be used to simulate the entire communication process from original image retrieval, through image compression, error coding, transmission, propagation over a channel, error detection and correction techniques (EDAC), image decompression, and final image display and evaluation.

In addition, image compression algorithms can be evaluated with respect to their performance when used for image transmission over a variety of communication channels. The results of these evaluations can then be used to suggest algorithm modifications which will improve performance over a communication channel.

The ICPIC testbed is modular in design and is built on the Signal Processing Worksystem (SPW) software developed by the Alta Group of Cadence Design Systems, Inc. and the Spread Spectrum Vulnerability Metric (SSVM) software developed by TASC. Use of the ICPIC testbed requires familiarity only with the material covered in the SPW tutorial manual. ICPIC expands the SPW analysis capabilities by providing the user with one standardized graphical interface with which to gain parallel access to all of the capabilities of both the SPW and SSVM module libraries.

ICPIC facilitates the construction of a complete simulation from analysis "primitives" by drag and drop operations. These primitives range in capability from low-level modules such as signal sources and simple scaler multipliers, to high-level modules that perform such capabilities as PSK (Phase Shift Keying) and EDAC (Error Detection and Correction). By utilizing the graphical interface along with the SPW and SSVM module libraries, ICPIC gives the user the capability to quickly construct an end-to-end simulation, allowing for rapid system design assessment and optimization.

The remainder of this section describes the image compression, EDAC, and channel propagation models available in ICPIC as well as the relationship between these components. Section 2 provides a detailed discussion of a complete end-to-end BPSK simulation, including both the image analysis and data preparation software tools which were developed for this effort and are available as part of the ICPIC testbed. Section 3 presents sample data outputs from the BPSK system. Section 4 presents a summary and recommendations for future analysis.

1.1 Image Compression

Image compression algorithms reduce storage and bandwidth requirements by reducing information redundancy in the transmitted image. The smaller the image, the less time the propagation channel has to degrade it by the addition of noise. Individual image compression algorithms are grouped under the higher level classifications of either Lossless or Lossy compression algorithms. Lossless means that the image data can be compressed and decompressed infinitely many times without suffering any loss in the information content. With lossy compression algorithms, there is some information loss every time the image is compressed and decompressed. The amount of loss depends on the compression factor, which is an arbitrary user-defined integer. Depending on the application, it may be acceptable to lose some information if it significantly decreases the transmission time while maintaining sufficient information content.

Depending on the specific image compression technique, certain portions of an image may be more sensitive to errors than other portions. The EDAC techniques can therefore be optimized for the more sensitive portions of the transmitted image. In general, errors in the image header need to be eliminated or at least minimized as the information contained in the header portion of the image contains the prescription (such as the horizontal and vertical resolution) for how to reconstruct the transmitted image from the received data.

To emphasize the need for image compression consider the following example. A digitized grey scale image consists of an array of picture elements (pixels) represented by numbers that correspond to the brightness (grey scale) of each pixel, with 0 = black and 255 = white. A grey-scale image has 1024 x 1024 such numbers taking integer values between 0 and 255. Thus the image is given by a matrix $(P_j)_{j \in \{0,...,1023\}^2}$ with $P_j \in \{0,...,255\}$. To encode this sample image would require a data file approximately one million bytes in size. Compression of this sample image with the appropriate image compression algorithm will facilitate the efficient and effective transmission of the image across a given communication system configuration.

For the ICPIC effort the following three image compression techniques were utilized: the Graphics Interchange Format (GIF), the Joint Photographic Experts Group (JPEG), and Wavelet. GIF is a popular commercial standard 8 bit/256 color protocol for transmission and interchange of raster graphic data in a way that is independent of the hardware used in it's creation or display. GIF is comprised of a lossless compression strategy based on LZW compression (Lempel-Ziv-Welch algorithm). The LZW technique was originally developed for compression of textual material, and compresses files by substituting commonly occurring character sequences with a reference to the first occurrence of the sequence. For fullscreen, 8 bit images of moderate

complexity, 4:1 compression is the average. Since this is a lossless compression scheme, the compression ratio can not be increased by trading off image quality.

GIF compression offers the following advantages and strengths:

- it is a lossless compression scheme for 8 bit images (since no image degradation occurs, repeated compression and decompression cycles are possible, also suitable for images where information loss can not be tolerated).
- it is ideally suited to stylized images such as line drawings, or those which contain only a limited number of colors (for these images it can produce good compression ratios).
- it is widely used and supported, with no runtime license required.

The weaknesses and disadvantages of GIF compression are as follows:

- it is not suitable for 24 bit images. When compressing such images, much of the color information is lost during the quantization process which reduces this to 8 bits. Good algorithms can however optimize this process so that the resultant image will still have good quality from a human point of view.
- the compression ratios are low. These can not be traded off against compression times or degree of loss of quality.
- it is not intended for moving images/video.
- it is not resolution independent.

JPEG is a lossy image compression scheme aimed at still images, in color or grey scale, and having some degree of complexity. The greater the compression ratio, the greater the degree of information loss. This can be user determined to optimize the trade-off between resultant image size and image quality. The JPEG compression algorithm exploits some of the ways in which the human eye perceives and analyzes images, so that compressed images still appear to be of high quality when looked at by human eyes. The total amount of time it takes to compress and decompress an image are symmetric, meaning that they take roughly the same amount of time.

The JPEG algorithm is based on the forward DCT (Discrete Cosine Transform), as applied to a block breakdown of the original image into 8 by 8 blocks, quantized down to a finite set of possible values, and then further transformed (run length encoding) and finally entropy encoded using Huffman or arithmetic coding.

The following list compares the compression ratios with the observed quality. A 20:1 compression ratio means that an image originally 900K will be compressed to 45K, which is one twentieth of its original size.

- 10:1 to 20:1 - High quality, with little or no observable loss in image quality to the human viewer.
- 30:1 to 50:1 - Moderate quality.
- 60:1 to 100:1 - Poor quality, suitable for thumbnails and previews. Marked blockiness and "Gibb's effect" occurs. The Gibb's effect is the name given to the phenomenon where disturbances and/or ripples can be seen at the margins of objects with sharp borders.

In comparison with GIF compressed images, high quality JPEG image compression produces an image 4 to 5 times smaller.

JPEG image compression offers the following advantages and strengths:

- it provides support for full 24 bit color images. In contrast, GIF compression only supports 8 bit images.
- the compressed image size and image quality trade-off can be determined by the user.
- it is ideally suited to images of real world scenes, or complex computer generated images.
- it is platform independent for displaying 24 bit images. It is currently the most widely adhered to image compression standard, with the algorithm, source code implementations, and public domain viewers readily available.

The weaknesses and disadvantages of JPEG are as follows:

- JPEG compression is a trade-off between the degree of compression, the resultant image quality and the amount of time required to compress and decompress an image. Also, blockiness results at the high image compression ratios.
- it produces poor image quality when compressing text or images containing sharp edges or lines (known as the Gibb's effect).
- it is not suitable for 2 bit black and white images.
- the degree of compression is greater for full color images than it is for grey scale images.
- it is not a suitable strategy for images that are still being edited, because every compression and decompression cycle causes a loss in information.

- it is not intended for moving images and/or video.
- it is not resolution independent. Does not provide for scalability, where the image is displayed optimally depending on the resolution of the viewing device.

Wavelet transforms are high compression algorithms capable of analyzing data at multiple resolutions (also known as "scale"). In other words, they cut up data into different frequency components, and then analyze each component with a resolution matched to its scale. In addition, transient events in the data are preserved by the analysis. When the wavelet transform is applied to a signal in the time domain, the result is a two dimensional, time-scale analysis of the signal. The transform of the data exhibits discrete steps in time on one axis, and discrete steps of resolution on the other. The wavelet transform has proven to be a useful tool for the compression and analysis of both signals and images. The discrete wavelet transform (DWT) is the wavelet transform as applied to a regularly sampled data sequence and the fast wavelet transform (FWT) is an efficient implementation of the DWT.

DWT has the distinct benefit of being able to perform the simultaneous localization of frequency and time. This gives the DWT the instantaneous advantage of information condensing. The wavelet transform reduces information in a signal roughly to averages and differences of neighboring pixels at discrete scale (frequency) levels. In an image, the information in areas of near constant color are collapsed into the average component and the difference components. Each average and difference component contains information about an area 4 times its size. If the differences are close to zero then the components can be discarded and all the information about the area will be contained in the average components.

The wavelet analysis procedure adopts a wavelet prototype function, called an analyzing wavelet or mother wavelet. In the FWT algorithm, the sampled data set is passed through scaling and wavelet filters. These filters are collectively known as a quadrature mirror filter (QMF) pair and are actually low-pass and high-pass filters which contain complementary bandwidths. The outputs of both filters are decimated (desampled) by a factor of two. The high-pass filtered data set is the wavelet transform detail coefficients at the same level of scale as the transform. The low-pass filtered data set is the approximation coefficients at the same level of scale. Temporal analysis is performed by using a high-frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet. Due to the decimation, both sets of coefficients have half as many elements as the original data set. The approximation coefficients can now be used as the sampled data input for another pair of wavelet filters, identical to the first pair, generating another set of detail and approximation coefficients at the next lower level of scale. This process can continue until the limit for the unit interval is reached. Because the

original signal or function can be represented in terms of wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using just the corresponding wavelet coefficients. Further compression is achieved by keeping only the most significant bits of the coefficients which is known as scaler quantization.

Threshold coding is the process whereby only those coefficients that exceed a specified threshold are retained. Threshold coding results in the sparse representation of the data which makes wavelets an excellent tool for data compression. Optimally, the coefficients are transmitted in order of decreasing size. The process of sending coefficients sequentially across a communication link to allow for the gradual reconstruction of the image is known as progressive transmission. Flexibility is a characteristic that makes the FWT an important tool for image transmission. The FWT lends itself ideally to applications which require reordering of digital information. Its adaptability is best seen in the progressive transmission of an image over a digital data link where incoming information builds on that already received, resulting in no computational or information overhead. In other words, the time and information it takes to decompress an image piecewise is the same as decompressing the image all at once.

As stated previously, the JPEG image compression method is based on a division of an image into blocks of 8 x 8 pixels, after which each block is transformed with a discrete cosine transform. While the JPEG method performs well at high or medium bit rates, it introduces perceptibly annoying blocking artifacts (due to block-wise transform coding) at low bit rates. The performance of JPEG deteriorates rapidly at compression ratios above 30:1. Due to good localization in both space and frequency domains of the basis function, the wavelet transform can obtain greater compression ratios thereby facilitating high data throughput which is required for real-time/video and image compression. As a result, the reconstruction quality of wavelet compressed images has moved well beyond the capabilities of the JPEG method.

1.2 Error Correction and Detection

When information is propagated between two points, errors are introduced due to noise along the propagation channel. Two things are therefore necessary; error detection, and (if possible) error correction.

The simplest form of error correction is to use an error-detecting code so conceived that any false or incorrect signal initiates a repetition of the transmitted character that was incorrectly received. This form increases not only the transmission time, but also the probability of signal detection and interception when this may be an issue of concern.

A better method is to divide the data into packets of fixed size, where each packet contains image information and a code that allows a user to detect, with high probability, if an error has been introduced. Based upon the known size of the data packet (N), and the number of bits of information (k), it is usually possible to remove the errors and reconstruct the original image.

SPW provides a number of EDAC algorithms, such as Reed-Solomon and Bose-Chaudhuri-Hocquenghem (BCH) which includes a vector BCH encoder and decoder. BCH encoding and decoding processes binary data in blocks. The convolutional encoder, which is often used with Viterbi decoders, typically works with arbitrarily long, continuous flowing bit streams. The Viterbi algorithms have also provided competitive performance with blocks of data. SPW provides both the hard and soft decision versions of the Viterbi decoding algorithm, with the soft decision version permitting substantially fewer residual errors. Viterbi decoding is the most commonly used form of trellis decoding, and requires substantially less storage of prior data than sequential decoding. Many satellite communications systems use Viterbi decoding.

Reed-Solomon encoding and decoding is well suited for use with M-ary signaling, and also for error distributions that are uniform in the symbol space. JTIDS uses Reed-Solomon coding. Included in the SPW software are vector Reed-Solomon coding and decoding and included in the ICPIC module libraries are M-ary modulation and demodulation to complement the Reed-Solomon coding modules.

In addition, TCM and V.32 encoders and decoders are available as part of the ICPIC module libraries. V.32 coding is an ISO standard for telephone line transmission.

1.3 Propagation Channel

The propagation channel describes the interaction between the signal and the environment between the transmitter and receiver. Among the channel models incorporated into ICPIC are noise, interference, tapped delay, and path loss.

The term background refers to additive contamination of the communication signal and includes both noise and interference. Noise is generally natural in origin and broadband, and includes thermal receiver noise and environmental or atmospheric noise due to natural electromagnetic effects. Interference is generally man-made and often narrowband. Table 1-1, Independent Parameters in Background and Channel Models, shows the occurrences and properties of the independent parameters used in background and channel components.

Independent Parameter	Occurrence	Relationships
Configuration center frequency	Random Interference HF Interference VHF Interference	Typically inside signal band
Configuration bandwidth	Random Interference HF Interference VHF Interference	Interferers outside detector input bandwidth have no effect on vulnerability
Configuration input file Name of file used to store random interference configuration	Custom Interference	Format same as configuration output file read by Custom Interference module
Configuration input file Name of file used to store random interference configuration	Random Interference	Format same as configuration output file read by Custom Interference module
Input SNR	Gaussian Noise Gamma Noise	Assumes only one noise source in use
Exponent describing power loss with propagation distance	Path Loss	
Frequency Resolution Spacing of grid of candidate interferer carriers	Random Interference	Less than configuration bandwidth
Minimum interferer width	Random Interference	Less than maximum interferer width
Maximum interferer width	Random Interference	Greater than minimum interferer width
Mean Power (dB above signal)	Random Interference	
Number of Interferers	Random Interference	Integer, at least one
Power standard deviation (dB)	Random Interference	Greater than one tenth of the mean interferer power
Scale factor Controls strength of Cauchy noise, which has infinite variance	Cauchy Noise	Input SNR is not defined
Spread Time interval between first and last taps	Tapped Delay	Less than interception collect time
Transmitter to communication receiver distance	Path Loss	
Transmitter to interception detector distance	Path Loss	

Table 1-1 Independent Parameters in Background and Channel Modules

ICPIC includes three zero mean, white noise sources: Gaussian, Gamma and Cauchy. The variance of discrete samples of the complex envelope of noise depends on the sampling frequency and power spectral density of the real noise, as shown in Appendix A. The Gaussian and Gamma noise modules are parametrized by the input SNR (Signal-To-Noise) ratio under the assumption that the two modules are used as alternatives, and not together.

The Gamma Noise module permits simulation of non-Gaussian effects and produces radically symmetric complex noise whose magnitude is a gamma variable. The marginal distributions of the real and imaginary components are not expressible as elementary functions, but decay exponentially. The only independent parameter is the input SNR and dB.

Cauchy noise represents heavy-tailed noise distributions and is an exceptional case, because it has infinite variance. The Cauchy Noise module produces radially symmetric complex noise whose real and imaginary components are marginally distributed as Cauchy random variables. The user may set a scaling parameter which multiplies a standard unit variable. The Cauchy variable has an undefined mean and infinite average power, so that signal to noise ratio has no meaning. Instead, Cauchy Noise is parameterized by a scaling factor.

The interference module provides the user with three options with which they may configure the module with an arbitrary number of interferers, with variable widths, powers and frequencies.

The Random option allows the user to specify the number of interferers, the mean and variance of the interferer powers, and the maximum and minimum interferer widths, and the maximum and minimum interferer carrier frequencies. A configuration of the interferers is then randomly generated from the specified distributions. The frequency and bandwidth of each interferer are generated from uniform distributions, and the power of the interferers are generated from a log-log distribution on analysis of empirical observations as given in Appendix B. The interferer width is implemented by BPSK (Binary Phase Shift Keying) modulating each interference carrier at a modulation frequency equal to half the interferer width.

The Custom option allows the user to directly specify the frequencies, widths, and powers of the interferers. The user is given the capability to specify the number of interferers; the configuration center frequency; the interferer center frequencies, expressed as offsets from the configuration center frequency; the interferer powers, in dB above the communication signal power; and the widths of the interferers. The bandwidth is implemented by BPSK modulating each interference carrier at a modulation frequency equal to half the bandwidth.

The HF and VHF options allow the user to select stored interference configurations typical of the HF and weak, medium and strong VHF bands. An interference background generated from a stored configuration chosen to be characteristic of typical operating environments is added to the signal. The user selects the center and width of the operating band, and the modules generate the interferers which lie in the selected band.

The channel category models phenomena which delay or distort the signal or change its power, such as multipath, propagation losses, and Doppler shifts.

A custom coded tapped delay line provides a foundation for channel modeling. The output is a linear combination of delayed versions of the input with constant weights specified by the user. The independent parameter is the total delay spread. The number of taps is equal to the product of the delay spread and the sampling frequency, and the user may specify arbitrary tap weights.

If the communication receiver and interferer are at different distances from the transmitter, the interferer will have a range advantage or disadvantage relative to the communicator. The Path Loss module models the loss with a power law decay of signal power as a function of propagation distance.

1.4 Interaction Between Compression, EDAC and Channel Model

EDAC techniques introduce delays due to processing, retransmission, and associated buffering, and each of these effects can have important system design implications. The designer must take into account how long an average transmission will take, the requirements for real-time image display, the tolerance on errors, transmit protocol, and how much buffering memory is available. EDAC effectiveness may be sensitive to the portion of the compressed image to which it is applied based upon the specific compression algorithm being used. For example, it may be necessary to completely eliminate errors from the image header. The errors themselves are a function of the noise model and it's magnitude. An accurate assessment of these combined effects is necessary in order to characterize system performance and optimize the system design. This is the functionality that the ICPIC workstation testbed provides.

2.0 An Example Of An End-To-End Simulation

This section describes how to set up an ICPIC simulation. The process is divided into SPW and non-SPW steps. The SPW steps involve building the actual simulation using the Block Diagram Editor (BDE) of SPW, and the non-SPW steps generate the input and analyze the output signal. The tradeoff between using the custom coded modules of the ICPIC environment or off-line analysis modules is discussed in Section 2.5.

2.1 Building The Simulation

Building a SPW simulation requires care in accounting for module-specific requirements. It is often possible to build the same simulation in slightly different ways that differ drastically in computational efficiency. Consider the simulation shown in Figure 1, which models the transmission of an image using BPSK modulation, BCH encoding, with the addition of White Gaussian Noise. This diagram is divided into eight sub-blocks for the purposes of this discussion. We will discuss each block and (where appropriate) possible alternatives and tradeoffs.

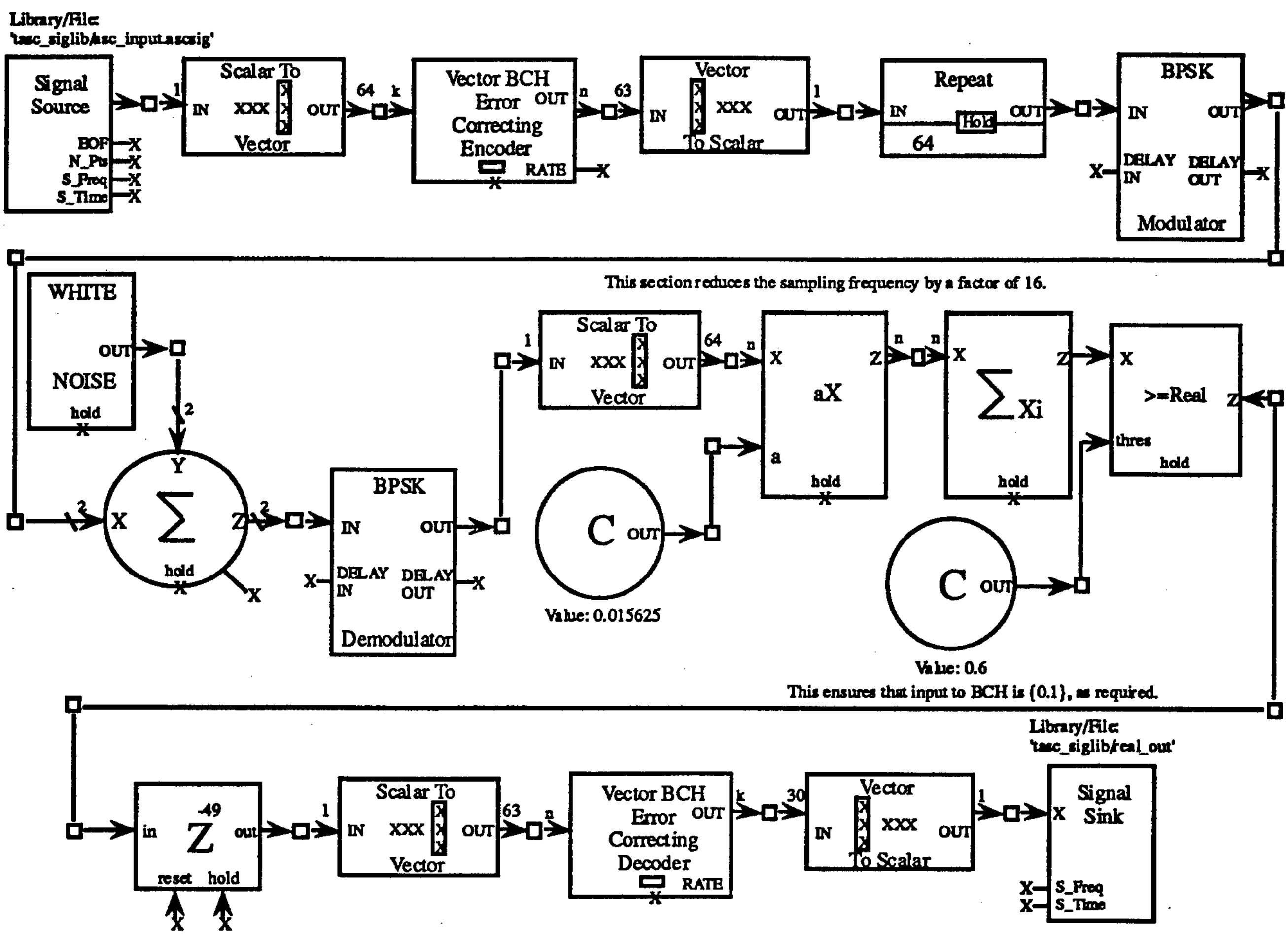

Figure 1: BPSK BCH Communication Model

Block 1: Signal Source

This defines the input signal, which is in SPW format. This is defined by a 12-line header and data, an example of which is shown in Figure 2. The details of this file and how it is generated from an image will be discussed in Section 2.2.

The Signal Source block, besides defining the input signal, displays it in the Signal Calculator module of the SPW.

```
$SIGNAL_FILE 9
$USER_COMMENT
$COMMON_INFO
SPW Version = 3.10
System Type = solaris2
Sampling Frequency = 16
Starting Time = 0
$DATA_INFO
Number of points = 9960
Signal Type = Double
$DATA
1
1
0
1
```

Figure 2: Sample SPW File

Block 2: Vector BCH Encoder

In this simulation we used a word size of N=63, and a data length of k=30, to give k/N close to 1/2. Contrary to the SPW documentation, the allowable values of k and N are not arbitrary. Figure 3 gives the allowable (k, N) pairs[1].

Since each block of 30 data points is being encoded into a 63-data length word, SPW requires that the Vector BCH Encoder block be fed in 30-data chunks. This is achieved with a *Scaler To Vector* block, which takes every 30 data points and transforms it into a 30 x 1 vector. The output from the BCH encoder is a 63 x 1 vector which is converted back into set of 63 scaler points with a *Vector To Scaler* module.

N	k
7	4
15	5, 7, 11
31	6, 11, 16, 21, 22
63	7, 10, 16, 18, 24, 30, 36, 39, 45, 51, 57

Figure 3: Allowable (N, k) Pairings For BCH Module

Alternative Block 2: Scaler BCH Encoder

This simulation originally attempted to use a scaler BCH encoder. This required that each data point be repeated 63 times. The result was an unacceptably long run time and storage requirements that were quickly exceeded by images files larger than 25 kbytes. These severe restrictions were removed by using the Vector BCH Encoder

It should be mentioned that the vector BCH encoder/decoder modules are not mentioned in the SPW Communication Library Reference.

Block 3: Repeat Block

The *Repeat* block takes each scaler input and repeats it the specified number of times (16 in this case). That is, if the sequence {1, 0, 1} is input into a *Repeat* block with a repeat factor of 3, the output would be {1, 1, 1, 0, 0, 0, 1, 1, 1}. Adding a Repeat block is equivalent to sending each data point a repeat factor number of times. This adds redundancy to the data and decreases the effect of channel noise by allowing it to be averaged out by Block 5. It was found that for S/N < 20, BCH encoding was not sufficient, by itself, to remove the effects of noise.

1 The SPW Communications Library Reference states that N is $2^m - 1$, where $2 < m < 9$ and $0 < k <= N$ is arbitrary within this range. In fact, k and N are restricted to the discrete pairings shown in Figure 3.

This brings up an important point about transmission frequency. The transmission frequency is not an explicit input parameter, but is effectively set to unity. As far as this simulation is concerned, it does not care what the frequency is, as there is no frequency-dependent module in this simulation. If we were to model the transmission frequency (typically 9,600 baud) explicitly, then each data point would have to be *Repeat*ed with a factor equal to the frequency. This would require prohibitively long simulation times and disk space requirements. We were therefore forced to model equivalent systems with manageable frequencies.

If the system being modeled operates at a frequency of v_Hz and a repeat factor of R, then the effective frequency is v/_R Hz since each data point needs to be transmitted R times.

Block 4: Modulator/Channel/De-Modulator

The encoded signal (consisting of zeros and ones) is BPSK-modulated and propagated through a channel. In this case, the channel is modeled as additive white Gaussian noise, with the addition performed by the Σ_module. Once propagated, the signal is then de-modulated and fed into an averager. The BPSK modulator/demodulator and the White Gaussian Noise blocks are custom-coded routines.[2]

Block 5: Noise Removal By Averaging

This block takes each segment of R-repeated data and averages it to get a single number. Remember that each block of R data points output from the decoder represents a single number; if noise were not present, then all these numbers would be identical. The average is performed by converting each block of R numbers into a vector, multiplying the vector by 1/R (with the **aX** block), and then summing the elements of the vector (Σx block). The constant 1/R is stored in the *Scaler Constant* block (C).

Block 6: Decision Block

Since the data consists of zeros and ones (as dictated by BPSK modulation), a check is performed to determine if the output from the averager exceeds some threshold. This threshold is stored in another constant block. If the average value exceeds this threshold then a one is output, if not, a zero is output. The threshold generally increases with decreasing S/N since noise in this simulation can only increase the signal strength. The actual value of the threshold is determined by trial and error.

2 Curiously, the SPW Communications Library has its own BPSK modulator, but not a BPSK demodulator module.

Block 7: Delay

The BPSK modulator/demodulator modules (Block 4) introduce a time delay into the signal. This must be accounted for in order that the BCH decoder (Block 8) be properly synchronized with the signal. Otherwise, the decoder will operate on segments from two separate signal code words (of size N), rather than a single code word. The size of the delay is N-1.

The delay can be seen by placing *Signal Sinks* before the BPSK modulator and after the BPSK demodulator (before and after Block 4) with a large S/N value in the *White Noise* module (or equivalently with the noise module disconnected). The two signals will appear identical, but phase-shifted. This test should always be done before using any ICPIC system configuration in order to insure an accurate baseline.

Block 8: Vector BCH Decoder

This block is analogous to Block 2, except the numbers are reversed since the signal is being decoded. The same comments mentioned in Alternative Block 2 apply here as well.

Block 9: Signal Sink

The Signal Sink is where the received signal is stored. From the Signal Calculator module of the SPW the received and input signals can be viewed and manipulated. For example, the difference signal, mean, error, and bit error rate can be determined using the functionality of the Signal Calculator.

2.2 Constructing The Input Signal

The input signal is constructed from an image file which can be in any format. The basic procedure is to read the image file, one bit at a time, and convert each bit into a binary representation. Since a bit is 8 bytes, each bit is converted into an 8-digit binary number. The program that does this is called **build_spw** and is invoked by typing:

build_spw ***image_file***

where *image_file* is the name of the image. The output is an ASCII file called **spw_in.asc,** which consists of a 12-line header followed by the binary-translated data. Each digit of each data point is placed in column 1 of each line after the header, as illustrated in Figure 2.

Once this file is created, it needs to be transferred to the appropriate SPW directory, which is called *local_path*/altadata/spwdata/signal_lib/, where *local_path* is the location of the top-level SPW directory, and *signal_lib* is the (arbitrary) name of the signal library where the user SPW signal files are stored. An example of *local*_path is /export/home/delmedico/. This transfer is accomplished by typing:

transfer

from the C-shell command line. **transfer** is a script that takes **spw_in.asc** and copies it to a file called **asc_input.asc** in the appropriate SPW directory. This script needs to be edited before using it the first time in order to modify *local_path* for your specific system.

2.3 Wavelet To SPW File Format Conversion

Since wavelet compression is not a standardized technique, and there are many different wavelet routines available, transforming a wavelet file to SPW format is a special case since we have to create the wavelet file in the first place. After much searching, we found the best package to use was **DSW**; this is a descendant of the David Sarnoff Labs wavelet package and is used by many government and private organizations. A limitation of **DSW** is that it only converts TGA image files to a wavelet representation. TGA is one of the earlier image storage formats that has the benefit of being easy to decode. To remove this limitation, we used the publicly-available **pbmplus** graphics file conversion package which allows conversion among most graphics file formats. **pbmplus** consists of a set of routines to convert from one format to another using intermediary formats. Using **bold** to represent executable programs, and unbold to represent file formats, the procedure for converting from GIF to wavelet and finally to SPW is:

GIF ==>**giftoppm** ==>PPM ==>**ppmtotga** ==>tga ==>**DSW** ==>wavelet file ==>**build** ==>SPW .

Alternatively, the starting point could have been TIFF or BMP, or another graphics format other than GIF. **pbmplus** does not have any JPEG conversion routines, so in this case we would use another public-domain program called Xview, which is invoked by typing **xv** at the command prompt. From here a JPEG file can be opened and saved in a GIF format and then converted using **gif2wav**. This entire process can be invoked with a script by typing:

gif2wav ***gif_filename***

where *gif_filename* is the name of the GIF file with the extension .gif assumed.

2.4 Data Analysis

The output signal can be analyzed within the Signal Calculator or from the UNIX command-line. From within the Signal Calculator you can;

1. Visually compare the input and output signals
2. Calculate and view the difference signal
3. Calculate statistics on the difference signal, such as mean, variance, and number of errors.

From the command line, a tool is provided to calculate all the quantities in Item 3 above, with the added convenience of batch mode processing. SPW provides a mechanism for batch mode processing a simulation when only one input parameter changes. In this way a user can parameterize the output statistics as a function of S/N, with all other variables kept constant. Each output is automatically stored in a separate file which can be parsed and fed into the command-line analysis code.

The code that calculates the output signal statistics is called rms and is invoked from the command line by

rms ***spw_output_filename***

where *spw_output_filename* is the output from SPW that is stored by Block 8 in Section 2.1.

The output from **rms** will be discussed in more detail in Section 3.

2.5 Reconstructing And Viewing The Transmitted Image

Although not a quantitative measure, the most intuitive way of judging how well the transmitted signal was received is to view the transmitted and received images side by side. This requires that the raw data from the *Signal Sink* (Block 8 in Section 2.1) be reconstructed to form an image file. This is accomplished by typing:

getdata

from the UNIX command line; this takes the ICPIC/SPW output file from the SPW signal library directory and places it in the user's working directory. As with the **transfer** script (Section 2.2), **getdata** needs to be edited before using it the first time in order to change the relevant path names. **getdata** takes the outfile file called **real.out** and copies it to the working directory as **spw.out.**

The data in **spw.out** is then reconstructed into an image file by typing:

build

from the UNIX command line. **build** takes each block of 8 lines past the 12-line header file, which consists of zeros or ones[3], treats it as an 8-bit binary number, and converts it back to an 8-bit decimal integer (0 - 255).

The analysis program (**rms**) uses the same procedure to convert from binary to decimal numbers. To do otherwise would ignore the implicit weighting of the binary digits. The outputs from **rms** are the number of bit errors (this is based on the binary representation), the RMS error between the original image and the output from ICPIC/SPW, and a quantity called the Contrast Error. This is basically a normalized RMS, where the normalization is the average pixel value of the image.

Once the image file is reconstructed, it can be displayed using **xv**. In some versions of **xv**, it is possible to display multiple files side by side. In other versions only a single file can be displayed; in this case **xv** must be invoked twice, once for the original and once for the reconstructed image. If the S/N ratio of the Gaussian noise is too low, then **xv** may not recognize the image as a valid format (it can handle most of them) and will not be able to display it.

3.0 Sample Output

The GIF image shown in Figure 4 was used to study the effects of varying the encoder parameters (k, N) and S/N. It has sufficient detail to show any deviations from the original and is small enough (19.2 Kb) to allow quick turn around times (under 10 minutes/run).

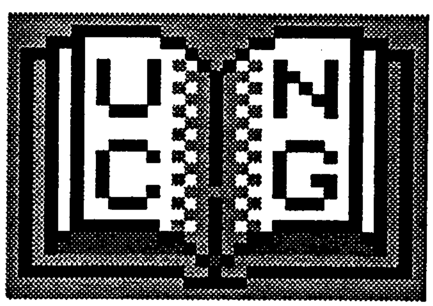

Figure 4: Book graphic test image

3 This is guaranteed by the Decision Block (Block 6 of Section 2.1).

Figure 5 shows plots of the Contrast Error for three (k, N) values. For all cases the transmitted image was identical to the original for S/N > 15 dB. For 7/63 coding (7 information bits in a code word of 63 bits), the transmitted image is identical to the original for S/N as low as 5 dB.

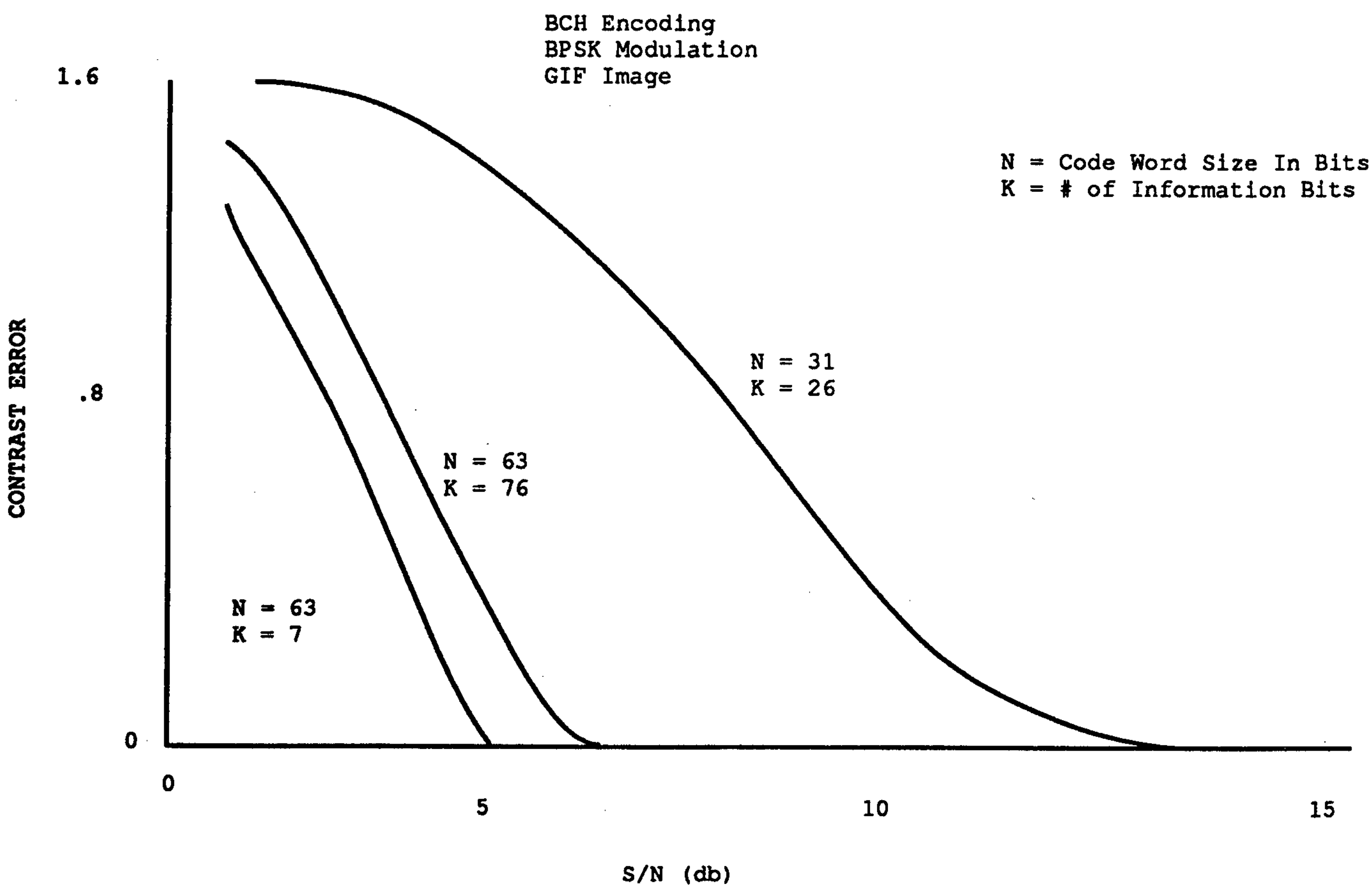

Figure 5: Variation Of Contrast Error With S/N

Notice that as S/N tends to zero, the error does not tend to infinity. The reason is that the decision block (Block 6) forces the input to the decoder to be zero or one. As such, as S/N decreases, the output saturates to all ones (totally black image). This represents the maximum deviation from the original image, and also points out that this deviation is image-dependent. For example, if the image had been mostly black, then the contrast error would have been smaller for the same system parameters. This same behavior was seen when looking at the fraction of incorrect bits in the received image.

Figure 6 shows the signal before and after Block 4, respectively, for the case of no noise. This illustrates the phase shift (ϕ) introduced by the BPSK modulator/demodulators, and illustrates the need for the delay module (Block 7). The delay is set to N - ϕ sample points. If the delay block is not added to the system, then the BCH decoder will receive segments from two consecutive code word blocks, rather than a single code block of 63 bits. An indication of an incorrect delay value is a deviation between the transmitted and original image when no noise is present. This test case should always be performed before attempting any tradeoff studies.

Figure 7 shows the signal before and after Block 4 with S/N = 5. The two signals are now clearly different, but all the deviations (except for the phase shift) are accounted for by the BCH decoder for k=7 and N=63.

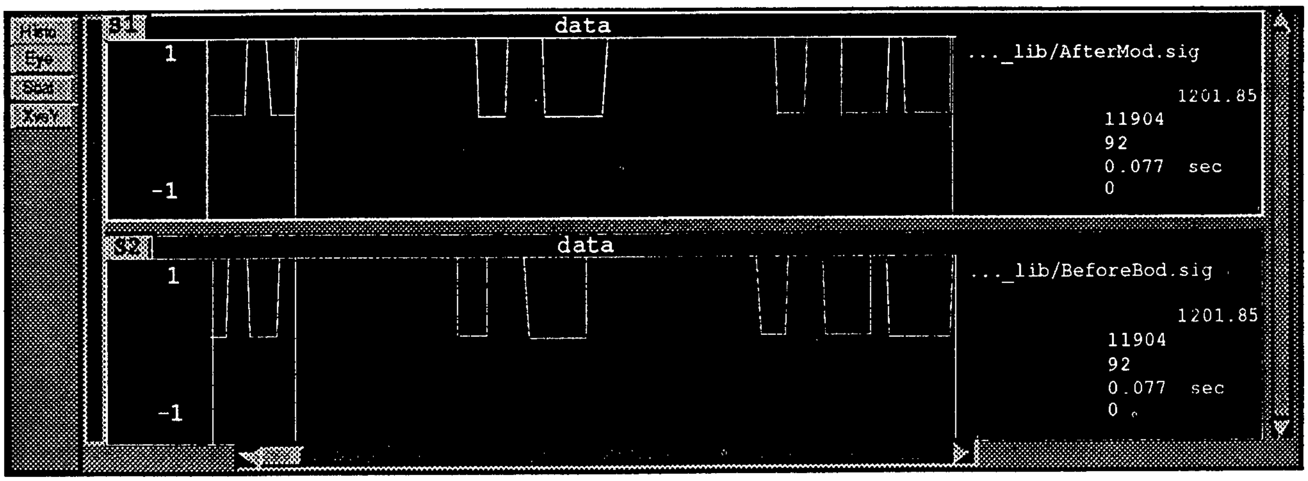

Figure: 6: Signal before (top) and after BPSK modulator block without noise. The signals are identicle, except for a phase shift.

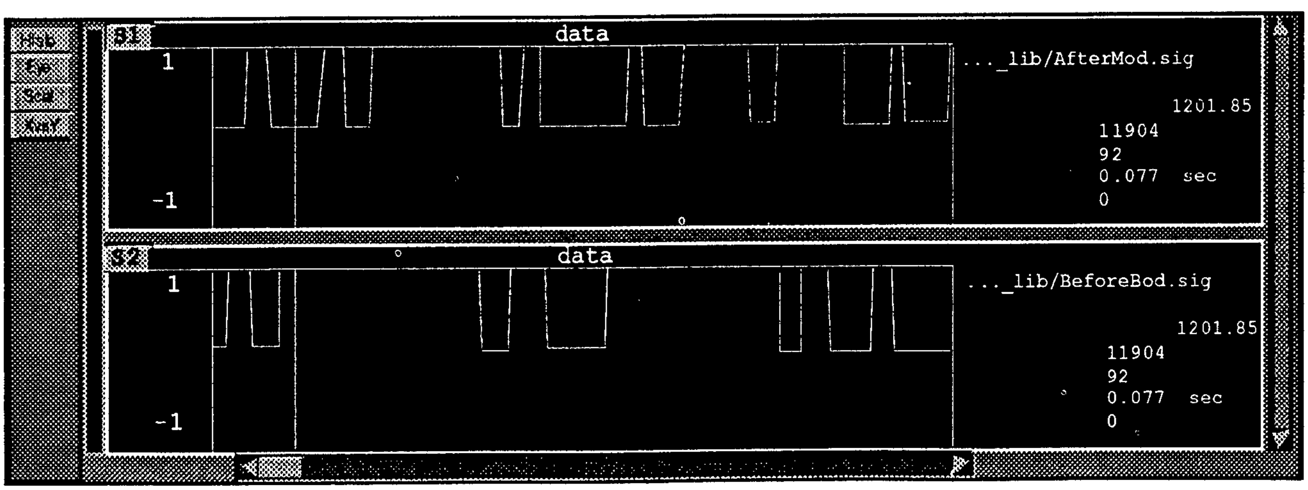

Figure: 7: Same as above, but with noise added (S/N = 5). The signals are now different, but this is corrected by the BCH decoder for K=7 and N=63.

4.0 Summary

ICPIC provides a flexible environment that gives the user the capability of modeling various communication channels, error detection and correction techniques (EDAC) and modulation types, as well as the ability to assess their effect on imagery compressed with various algorithms. Software tools are provided to convert any type of image (including wavelet) into a SPW format, and to convert from one image format to another. Changing the channel and EDAC models is accomplished by dragging and dropping operations in the BDE (Block Diagram Editor) and then setting the appropriate parameters for each module.

The image preparation and analysis tools were developed off-line from ICPIC/SPW to run from the UNIX command line. This was found to be a much more efficient approach then to force everything within the ICPIC/SPW framework. Included in these off-line software tools is the capability to reconstruct the received image from the ICPIC/SPW output. Depending on the errors in the received image, graphics viewers (such as **xv**) may not be able to display the reconstructed image. This is especially true if errors occur in the header file. One approach to minimize header errors is to agree on a particular fixed header, and then only send the actual data. It is also possible to develop robust image viewers that attempt to correct for corrupt data. These issues should be investigated in the future.

To date ICPIC's analysis capabilities have been applied to simplex communications channels. These are one-way channels where the receiver does not have the capability to request re-transmission of data missed due to channel bit-errors. Simplex channels are primarily used today in the tactical arena. These are situations in which the receiver does not want to turn on a transmitter out of fear of detection by hostile forces or alternatively for portability reasons the receiver sight lacks sufficient power for transmission. UHF satellite channels, such as FLTSAT, are a good example of channels used primarily for simplex communications. The application of the developed capability to evaluate the performance on compressed imagery of channels such as these could also be an area for future investigation.

Another area for investigation is how to specify quality of service and to best evaluate the results of channel induced corruption on compressed imagery. Quantitative comparison (RMS error, etc.) of the reconstructed corrupted images and original images can often be misleading. Qualitative comparison by side-by-side viewing of the reconstructed corrupted image and original image is often more useful but is subjective. The basic problem is that of objectifying a basically subjective phenomena. In addition, the performance depends on the particular scenario. For example, is the goal to pick out a rocket launcher in the woods, or to distinguish between classes

of tanks? It may be acceptable to use much higher S/N ratios to answer the first question and this may effect the final choice of system operating parameters.

The fact that extremely corrupted images cannot be decompressed at all for viewing leads to yet another area for future investigation. Since an image that cannot be decompressed is useless, it is desirable to conduct more detailed tradeoff studies with more complicated channel models, better imagery and different compression algorithms in order to investigate the error threshold for "decompressibility" for different EDAC methods and compression algorithms. This could be accomplished for various channels and modulation types. The goal would be to document which EDAC methods, modulation types and compression algorithms are best suited to getting a decompressible (and hopefully viewable) image through under various channel conditions.

These are significant issues that need to be fully addressed, and ICPIC/SPW provides the framework to address them.

Appendix A
Complex, Sampled, White Noise

Waveform simulations generally process discrete samples of continuous waveforms. In the case of white noise a continuous waveform is not available to sample, so random discrete sequences are generated directly, and a method is needed to set the parameters of the generating probability distribution. The chosen method is to require the discrete simulation of transmitted power through any filter to match exactly the transmitted power in the corresponding continuous process.

Complex envelopes are processed so that real white noise, which cannot be expressed in terms of quadrature components, are modeled as real bandpass white noise n with a power spectral density given by

$$-S_n(f) = \begin{cases} S_0 & \left||f|-f_{ces}\right|<B/2 \\ 0 & otherwise \end{cases} \tag{A-1}$$

where f_{ces} is the signal center frequency, and B is much larger than the largest bandwidth being simulated, but less than $2f_{ces}$. The complex envelope $z(t)$ of $n(t)$ relative to f_{ces} has a power spectral density given as

$$S_z(f) = \begin{cases} 2S_0 & |f|<B/2 \\ 0 & otherwise \end{cases} \tag{A-2}$$

and the quadrature components x and y have identical power spectra:

$$S_x(f) = S_y(f) = S_z(f) \tag{A-3}$$

Each quadrature component is modeled by a discrete random sequence with variance determined by the criterion that the noise power transmitted through any real lowpass filter acting as the noise baseband can be accurately modeled by the discrete simulation. If the quadrature component x is passed through a filter with transfer function $H(f)$ and real impulse response function $h(\tau)$, the transmitted power is:

$$P = E\left\{\left(\int_{-\infty}^{\infty} x(t-\tau)\,h(\tau)\,d\tau\right)^2\right\} \tag{A-4}$$

Expanding the square as the product of two integrals and taking the expectation of the integrand gives:

$$P = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R_x(\tau_2 - \tau_1)\, h(\tau_1)\, h(\tau_2)\, d\tau_1 d\tau_2 \qquad \text{(A-5)}$$

where $R_x(\tau)$ is the autocorrelation function of x. Expressing $R_x(\tau)$ as the Fourier transform of S_x and using (A-2) and (A-3) gives:

$$P = 2S_0 \int_{-\infty}^{\infty} |H(f)|^2\, df \qquad \text{(A-6)}$$

In a discrete simulation (A-5) becomes:

$$P_{discrete} = f_{sa}^{-2} \sum_{i,k} R_x\left(\frac{i-k}{f_{sa}}\right) h\left(\frac{i}{f_{sa}}\right) h\left(\frac{k}{f_{sa}}\right) \qquad \text{(A-7)}$$

which simplifies, since the discrete random numbers are uncorrelated, to

$$P_{discrete} = f_{sa}^{-2} \sigma_x^2 \sum_i h^2\left(\frac{i}{f_{sa}}\right) \qquad \text{(A-8)}$$

where

$$\sigma_x^2 = R_x(0) \qquad \text{(A-9)}$$

is the variance of the discrete time sequence modeling the x-component. For large sample frequency the sum in (A-8) approximates an integral, giving:

$$P_{discrete} \cong f_{sa}^{-1} \sigma_x^2 \int_{-\infty}^{\infty} |h(\tau)|^2\, d\tau \qquad \text{(A-10)}$$

so using the norm preserving property of Fourier transforms gives:

$$P_{discrete} \cong f_{sa}^{-1} \sigma_x^2 \int_{-\infty}^{\infty} |H(\tau)|^2\, df \qquad \text{(A-11)}$$

Comparing (A-11) to (A-6) and requiring the discrete simulation to model the continuous process accurately means the variance of the discrete quadrature component must be given by:

$$\sigma_x^2 = 2f_{sa}S_0 \tag{A-12}$$

while (A-3) shows both quadrature components have the same power so,

$$\sigma_x^2 = \sigma_y^2 = 2f_{sa}S_0 \tag{A-13}$$

The intended communicator's input signal-to-noise ratio is a dimensionless quantity that is of greater interest to the user than noise variances of power spectral densities. For communication signal power equal to unity shows the input signal-to-noise ratio is given by:

$$(S/N)_i = \frac{1}{f_{da}S_0} \tag{A-14}$$

Combining (A-13) with (A-14) gives:

$$\sigma_x^2 = \sigma_y^2 = \frac{2f_{sa}}{f_{da}(S/N)_i} \tag{A-15}$$

which is built into the white Gaussian noise block so the independent parameter set by the user is the communicator's input signal-to-noise ratio.

Appendix B
Interference Power Distributions

Interference environments change dramatically with geographical location, season, time of day, and frequency band, and while trends have been established [1], many of the details of the environments remain unpredictable. To model the multiplicity of unpredictable environments a Random option is available which randomly generates a configuration of interferers, subject to probability distributions whose parameters are specified by the user. Empirical observations reported and analyzed suggest the use of a specific distribution of interferer powers. Simplifying the distribution and parameterizing in terms of the mean and variance of the interferer powers enhances the Random option's ease of use.

The probability distribution used is

$$F(p) = \begin{cases} 0 & p < p_{\min} \\ 1 - Cp^{-d} & p_{\min} < p < p_{\max} \\ 1 & p > p_{\max} \end{cases} \tag{B-1}$$

and gives the probability that a randomly chosen interference line will have power less than p. Interferer powers are measured in units of the signal power at the communication receiver, so p is the ratio of interferer power to signal power. For simplicity $P_{\max}$ will be taken to approach infinity, and $P_{\min}$ will be chosen so that *F(p)* approaches zero continuously at $P_{\min}$, which requires $1 - Cp_{\min}^{d} = 0$, so

$$P_{\min} = C^{1/d} \tag{B-2}$$

Differentiating (B-1) gives the probability density as

$$f(p) = \begin{cases} 0 & p < p_{\min} \\ Cdp^{-d-1} & p > p_{\min} \end{cases} \tag{B-3}$$

The mean power is given by

$$\mu = \int_{p_{\min}}^{\infty} pf(p)\,dp \tag{B-4}$$

and using (B-3) and evaluating the integral gives:

$$\mu = \frac{Cd}{d-1} p_{\min}^{1-d} \tag{B-5}$$

provided

$$d - 1 > 0 \qquad \text{(B-6)}$$

otherwise the density in (B-3) has infinite mean. The variance is given by

$$\sigma^2 = \int_{p_{\min}}^{\infty} p^2 f(p)\, dp - \mu^2 \qquad \text{(B-7)}$$

and using (B-3) and evaluating the integral gives:

$$\sigma^2 = \frac{Cd}{d-2} p_{\min}^{2-d} - \mu^2 \qquad \text{(B-8)}$$

provided

$$d - 2 > 0 \qquad \text{(B-9)}$$

otherwise the density in (B-3) has infinite variance. Using (B-2) to eliminate $p_{\min}$ from (B-5) and (B-8) gives

$$\mu = \frac{d}{d-1} C^{1/d} \qquad \text{(B-10)}$$

and

$$\sigma^2 = \frac{d}{d-2} C^{2/d} - \mu^2 \qquad \text{(B-11)}$$

Solving for the parameters C and d in terms of the mean and variance proceeds by using (B-10) to eliminate C from (B-11) giving:

$$d^2 - 2d - \frac{\mu^2}{\sigma^2} = 0 \qquad \text{(B-12)}$$

and taking the positive root gives:

$$d = 1 + \sqrt{1 + \frac{\mu^2}{\sigma^2}} \qquad \text{(B-13)}$$

and rearranging (B-10) gives:

$$C = \left(\frac{(d-1)\mu}{d} \right)^d \tag{B-14}$$

Comparing (B-13) with (B-9) shows that as the standard deviation of the distribution becomes much larger than the mean $\mu^2 / \sigma^2 \to 0, d \to 2$ and the distribution approaches a singular limit.

The user supplies the mean power E_p and the power standard deviation s_p in dB so

$$\mu = 10^{E_p/10} \tag{B-15}$$

and

$$\sigma = 10^{E_p/10} \tag{B-16}$$

(B-15) and (B-16) in (B-13) and (B-14) are used to supply the coefficient values to the distribution in (B-1), and then randomly generate the interferer powers.

Appendix C
ICPIC Modules

The following list of entries uses a format based on the Unix man pages, this format provides easily accessible documentation of the modules included in the ICPIC workstation testbed. The modules are ordered alphabetically by component categories. The entry for each module provides a functional description of each module, the path leading to the module when using the SPW module selection under the "logical grouping" option, the input and output ports, and the parameters in the module.

There are two fundamental types of modules, known in SPW parlance as "custom-coded" modules and "multi-level" modules. The former refers to modules developed from scratch in C code. These are designated in the documentation by "Coded in C". Modules not indicated as such are multi-level. These are typically subsystems built from lower-level modules such as SPW primitives, custom-coded modules, or even lower level multi-level modules.

CONTENTS

8Ary DS SPREADER

DESCRIPTION

Multiplies the complex input by an 8-fold complex pseudonoise sequence with a chip rate specified by the user as a multiple of the data rate. If the input signal has constant unit magnitude, the output power is 1. Delays the output signal by the input delay value for automatic synchronization. Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Direct Sequence > 8ary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nch	Number of chips per symbol
fda	Data rate
sfreq	Sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

walib/8ary_ds

8PSK DEMODULATOR

DESCRIPTION

An 8-ary PSK demodulator.

MENU PATH

Communicator > Demodulator > 8PSK

INPUT PORT

IN	Complex
DELAY_IN	Real

OUTPUT PORT

OUT	Real
DELAY_OUT	Real

PARAMETERS

fda

sfreq

LIBRARY / FUNCTION

modlib/8psk-demod

8PSK MODULATOR

DESCRIPTION

An 8-ary PSK modulator. A binary input stream is gray coded onto an eight phase shift keyed signal

MENU PATH

Communicator > Modulator > 8PSK

INPUT PORT

IN	Real
DELAY_IN	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/8PSK-mod

8PSK SOURCE

DESCRIPTION

Outputs a narrowband complex envelope resulting from interleaving a pseudorandom bit source, and using the result as the data stream for an 8-ary phase shift keying modulation. The user sets the numbers of columns and rows to define the interleaving, and accepting the default value of 1 for both produces a non-interleaved data stream.

MENU PATH

Communicator > Modulated Source > 8PSK

INPUT PORT

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

ncols	Interleaver columns
nrows	Interleaver rows
fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/8psk-source

ADAPTIVE THRESHOLD EXCISION CORE

DESCRIPTION

Combines the two input vectors to form a complex vector, and removes the components with squared magnitude greater that an exceedance factor times a mean square magnitude. The mean square magnitude is of all the bins except a number of strongest bins. The user specifies the length of the vectors, the number of strongest bins to be excluded from the calculation of mean square magnitude, and the exceedance factor. The spectral resolution, or bin size is the sampling frequency divided by the dimensionality. The algorithm recomputes the threshold for each successive block of data and so adapts to changing background levels.

MENU PATH

None

INPUT PORTS

X_re	Real vector
X_im	Real vector

OUTPUT PORT

Y_re	Real vector
Y_im	Real vector

PARAMETERS

blocksize	Number of points used in the FFTs
nexempt	Number of points excluded in the calculation of mean square magnitude
exceedfactor	Exceedance factor
window_type	Type of window used in the forward FFT
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exadapt_cmp

Coded in C

ADAPTIVE WHITENING FILTER

DESCRIPTION

Implements a non-overlapped version of the adaptive whitening matched filter, which serves as the basis for the mmse adaptive whitening filter.

MENU PATH

None

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

blocksize	Length of FFT
average_type	Periodogram averaging type, either 'sliding_block' or 'decaying'
averaging_block	Length of sliding window for 'sliding_block' averaging type
daniell_param	Width of frequency domain smoothing window
alpha	Decay parameter for 'decaying' averaging type
window_type	Tapering window applied to input blocks to provide additional smoothing
sfreq	Sampling frequency

LIBRARY / FUNCTION

prlib/adapt_whiten

AGILITY

DESCRIPTION

Produces trigger pulses separated in time by random intervals called blocks. The user specifies the upper and lower bounds of a uniform distribution; each sample of the uniform distribution is rounded to the nearest multiple of the data symbol duration; and the result is used as the current block length. Setting an integer flag to 1 stores the trigger pulses and block durations as signals for subsequent inspection in SDE.

MENU PATH

Spread Spectrum > Agility

INPUT PORT

IN	None

OUTPUT PORT

TRIGGER	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
sfreq	Sampling frequency
fda	Data rate
store_agility	1 to store trigger pulses and block lengths, 0 otherwise

LIBRARY / FUNCTION

walib/agility

ANALOG TO VECTOR CONVERSION

DESCRIPTION

Coded in C

MENU PATH

Utilities > Vector > Analog to Vector

INPUT PORT

IN	Real
HOLD	Real

OUTPUT PORT

OUT	Real Vector

PARAMETERS

OUT_IOVEC_LEN

Output vector length

LIBRARY / FUNCTION

utlib/real_to_vector

ANNOUNCE

DESCRIPTION

Produces a message in a pop-up window at the completion of a simulation run.

MENU PATH

Utilities > Input/Output > Announce

INPUT PORT

HOLD	Real

OUTPUT PORT

None

PARAMETERS

control	Print message if > 0
message_1	First message line
message_2	Second message line

LIBRARY / FUNCTION

utlib/announce

Coded in C

AUTO STEPPER

DESCRIPTION

Produces a stepped output which cycles through a set of values specified by the user. The values are regularly spaced and specified by the starting value, the increment, and the number of values. The user controls the dwell time, which is the time interval between changes in the output.

MENU PATH

Utilities > Sources > Auto Stepper

INPUT PORT

HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

dwell_time	Time interval between changes in output
start_freq	Initial output value
step_freq	Step in output value
no_of_values	Number of output values
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/auto_stepper

BANDPASS CYCLIC FEATURE DETECTOR SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form a signal present input to one detector and a signal absent input to a second detector. The detectors are each cyclic feature detectors for bandpass cyclic frequencies. The result of processing the signal present case is the H_1 output and the result of processing the signal absent case in the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Cyclic Feature > Bandpass, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORT

H1	Complex
H0	Complex

PARAMETERS

cyclic_offset	Offset of cyclic frequency from twice signal center frequency
filter_zone	Spectral frequency zone: 0 for lowpass, 1 for bandpass
lp_edge	Spectral lowpass edge frequency
bp_center	Spectral bandpass center frequency
bp_bw	Spectral and pass bandwidth
delay_time	Delay in delay and multiply section of detector systems
collect_time	Output integration time in detector systems
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise

BANDPASS CYCLIC FEATURE DETECTOR SYSTEM, continued

sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	Data rate

LIBRARY / FUNCTION

delib/cyc_feat2

BANDPASS CYCLIC FEATURE DETECTOR CORE

DESCRIPTION

Simulates the action of a cyclic feature detector for bandpass cyclic frequencies with complex envelope input. Provides optional storage of intermediate signals.

MENU PATH

Interceptor > Detector > Cyclic Feature > Bandpass, core

INPUT PORT

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

cyclic_offset	Offset of cyclic frequency form twice signal center frequency
filter_zone	Spectral frequency zone: 0 for lowpass, 1 for bandpass
lp_edge	Spectral lowpass edge frequency
bp_center	Spectral bandpass center frequency
bp_bw	Spectral bandpass bandwidth
delay_time	Delay in delay, conjugate and multiply section
collect_time	Output integration time
sfreq	Sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/cyc_feat2_core

BANDPASS CYCLIC SPECTRUM

DESCRIPTION

Computes the bandpass component of the cyclic spectrum of a complex signal in SDE. The cyclic frequency of the bandpass cyclic spectrum computed is entered as an offset from twice the assumed carrier frequency. The spectrum is computed for spectral frequencies between $-f_{sa}/2$ and $f_{sa}/2$ where f_{sa} is the sampling frequency.

MENU PATH

SDE: SYSTEM/MACRO/EXECUTE/vmlic/Cyclic_bandpass

PARAMETERS

numsig	Display number of signal to analyze
alpha	Cyclic frequency desired
samp_freq	Sampling frequency
numpts	Number of samples to use in the computation
numffpts	Number of points in the FFT (must be a power of two)

LIBRARY / FUNCTION

spwdata/sdemacro/vmlib/Cyclic_high.mac

BANDPASS INTEGRATOR

DESCRIPTION

Multiplies the input by a complex tone with frequency equal to the bandpass integrator center frequency, and averages the result over the integration time.

MENU PATH

Utilities > Signal Processing > Bandpass Integrator

INPUT PORT

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fcebi	Center Frequency
inttime	Integration time
sfreq	Sampling Frequency

LIBRARY / FUNCTION

utlib/bp_integrator

BINARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a binary complex pseudonoise sequence with a chip rate specified by the user as a multiple of the data rate. If the input signal has constant unit magnitude, the output power is 1.

MENU PATH

Spread Spectrum > Direct Sequence > Binary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nch	Number of chips per symbol
fda	Data rate
sfreq	Sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

walib/binary_ds

BIT ERROR ESTIMATOR

DESCRIPTION

Compares the input bit stream to a synchronized reference data stream on the global connector called 'data', and computes the fraction of the total number of input bits which do not match the reference. The module writes the result to a file with a name selected by the user.

MENU PATH

Communicator > Metrics > Bit Error Rate

INPUT PORT

BITS OUT	Real
DELAY_IN	Real

OUTPUT PORT

None

PARAMETERS

filename	File name
fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

vulib/bit_error_rate

BPSK DEMODULATOR

DESCRIPTION

Demodulates a binary PSK waveform, producing a binary sequence.

MENU PATH

Communicator > Demodulator > BPSK

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Real

PARAMETERS

fda	Bit rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

modlib/bpsk_demod

BPSK MODULATOR

DESCRIPTION

A binary PSK modulator.

MENU PATH

Communicator > Modulator > BPSK

INPUT PORT

IN	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/bpsk_mod

BPSK SOURCE

DESCRIPTION

Outputs a narrowband complex envelope resulting from interleaving a pseudorandom bit source, and using the result as the data stream for a binary phase shift keying modulation. The user sets the numbers of columns and rows to define the interleaving, and accepting the default value of 1 for both produces a non-interleaved data stream.

MENU PATH

Communicator > Modulated Source > BPSK

INPUT PORT

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

ncols	Interleaver columns
nrows	Interleaver rows
fda	Bit rate
sfreq	Sampling rate

USAGE EXAMPLES

SSVM>Demonstrations> Correlation Radiometer
SSVM>Demonstrations> Scanner

LIBRARY / FUNCTION

modulib/bpsk_source

BURST

DESCRIPTION

Reduces the duration of the transmitted signal from the collect time to the burst duration

MENU PATH

Spread Spectrum > Burst

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Real

PARAMETERS

duration	Scale parameter
collect_time	Collect time
sfreq	Sampling frequency

LIBRARY / FUNCTION

walib/burst

CAUCHY NOISE

DESCRIPTION

Produces radially symmetric complex noise whose real and imaginary components are marginally distributed as Cauchy random variables. The user may set a scaling parameter, which multiplies a standard unit variable. The Cauchy variable has an undefined mean and infinite average power, so that signal to noise ratio has no meaning. If the scaling parameter is α, the marginal probability densities of the real and imaginary parts are given by

$$\frac{1}{\pi\alpha}\frac{1}{1+(x/\alpha)^2}$$

MENU PATH

Background > Noise > Cauchy

INPUT PORT

None

OUTPUT PORT

OUT	Complex

PARAMETERS

mult	Scale parameter

LIBRARY / FUNCTION

bklib/cauchy_noise

CENTER FREQUENCY MISMATCH

DESCRIPTION

Accounts for the difference, if any, between the center frequency of the communication signal and the center frequency of the receiver.

MENU PATH

Utilities > Signal Processing > Center Freq Mismatch

INPUT PORT

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fces	Signal center frequency
fced	Receiver center frequency
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/cent_freq_mm

CHIP RATE AGILITY

DESCRIPTION

Outputs trigger pulses at a rate which varies pseudorandomly from one block of time to the next, but is constant within each block.

MENU PATH

Spread Spectrum > Chip Rate Agility

INPUT PORT

None

OUTPUT PORT

OUT	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
nchav	Average number of chips per data symbol
hw	One sided percent chip length fluctuation
sfreq	Sampling frequency
fda	Data rate
Store_agility	1 to store trigger pulses and block lengths, 0 otherwise

LIBRARY / FUNCTION

walib/chip_rate_agty

CHIP RATE AGILE 8ary DS SPREADER

DESCRIPTION

Multiplies the complex input by an 8ary complex pseudonoise sequence, which has a rate that remains constant over blocks of time and varies pseudorandomly between blocks. The chip rate agile DS spreaders use the Chip Rate Agility module to trigger the generation of the M-ary symbols used in spreading. The chip rates result from sampling a uniform distribution and rounding to the nearest multiple of the data rate. The Chip Rate Agility module uses the Agility module to trigger generation of new blocks. Block lengths vary pseudorandomly, and are rounded to the nearest multiple of the data symbol duration.

The user specifies the minimum and maximum block lengths; the average chip rate as a multiple of the data rate; the maximum percentage one sided chip rate fluctuation; and whether the block lengths, block boundary pulses, instantaneous chip rate, and chip boundary pulses are stored. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization. Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Agile Direct Sequence >8ary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
nchav	Average number of chips per data symbol
hw	One sided percent chip length fluctuation
store_agility	1 to store block lengths, block boundary pulses, chip rate and chip boundary pulses; 0 otherwise
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

walib/a_8ary_ds

CHIP RATE AGILE BINARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a binary complex pseudonoise sequence, which has a rate that remains constant over blocks of time and varies pseudorandomly between blocks. The chip rate agile DS spreaders use the CHIP RATE AGILITY module to trigger the generation of the M-ary symbols used in spreading. The chip rates result from sampling a uniform distribution and rounding to the nearest multiple of the data rate. The CHIP RATE AGILITY module uses the AGILITY module to trigger generation of new blocks. Block lengths vary pseudorandomly, and are rounded to the nearest multiple of the data symbol duration.

The user specifies the minimum and maximum block lengths; the average chip rate as a multiple of the data rate; the maximum percentage one sided chip rate fluctuation; and whether the block lengths, block boundary pulses, instantaneous chip rate, and chip boundary pulses are stored. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization. Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Agile Direct Sequence > Binary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
nchav	Average number of chips per data symbol
hw	One sided percent chip length fluctuation
store_agility	1 to store block lengths. block boundary pulses, chip rate and chip boundary pulses; 0 otherwise
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

walib/a_binary_ds

CHIP RATE AGILE QUATERNARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a binary complex pseudonoise sequence, which has a rate that remains constant over blocks of time and varies pseudorandomly between blocks. The chip rate agile DS spreaders use the CHIP RATE AGILITY module to trigger the generation of the M-ary symbols used in spreading. The chip rates result from sampling a uniform distribution and rounding to the nearest multiple of the data rate. The CHIP RATE AGILITY module uses the AGILITY module to trigger generation of new blocks. Block lengths vary pseudorandomly, and are rounded to the nearest multiple of the data symbol duration.

The user specifies the minimum and maximum block lengths; the average chip rate as a multiple of the data rate; the maximum percentage one sided chip rate fluctuation; and whether the block lengths, block boundary pulses, instantaneous chip rate, and chip boundary pulses are stored. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization. Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Agile Direct Sequence > Quaternary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
nchav	Average number of chips per data symbol
hw	One sided percent chip length fluctuation
store_agility	1 to store block lengths. block boundary pulses, chip rate and chip boundary pulses; 0 otherwise
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

walib/a_quatary_ds

CHIP RELEASE

DESCRIPTION

Outputs trigger pulses at a rate equal to the input signal. The output pulses result from differentiation a square wave, which in turn is formed by thresholding a sinusoid with frequency equal to the input signal. The frequency is not rounded to synchronize with the interval between simulation iterations.

MENU PATH

Utilities > Sources > Chip Release

INPUT PORT

FREQ	Real

OUTPUT PORT

PULSES	Real

PARAMETERS

sfreq	Sampling frequency

LIBRARY / FUNCTION

walib/chip_release

COMMUNICATION COUPLER

DESCRIPTION

Adds the transmitted signal and background noise to form the signal received by the communication receiver.

MENU PATH

Channel > Couplers > Communicator

INPUT

SIGNAL	Real
BACKGROUND	Real

OUTPUT

SIG+NOISE	Real

PARAMETERS

None

LIBRARY / FUNCTION

chlib/comm_coupler

COMMUNICATORS COMPOSITE PREPROCESSOR

DESCRIPTION

Lowpass filters the input signal then optionally applies a frequency based excision technique.

MENU PATH

Communicator > Preprocessor > Composite

INPUT PORTS

IN	Complex
DELAY_IN	Real

OUTPUT PORTS

OUT	Complex
DELAY_OUT	Real

PARAMETERS

f_c	Filter cutoff frequency
delta_f	Filter transition region width
a_r	Filter stopband attenuation
sfreq	Sampling frequency
index	Selection index (0-no excision, 1-fixed, 2-Adaptive, 3-Whitening)
store_spectra	Store spectra? (index > 0)
blocksize	Number of points used in the FFT's (index > 0)
nlargest	Number of bins to be excised (index = 1)
exceedfactor	Exceedance factor (index = 2)
nexempt	Number of points excluded in the calculation of the mean square magnitude (index = 2)

LIBRARY / FUNCTION

prlib/communication

COMMUNICATOR'S ADAPTIVE THRESHOLD EXCISION SYSTEM

DESCRIPTION

Performs a spectral analysis of the input and removes the components with squared magnitude greater than an exceedance factor times a mean square magnitude. The mean square magnitude is of all the bins except a number of strongest bins. The user specifies the size of the FFT's, the number of strongest bins to be excluded from the calculation of mean square magnitude, the exceedance factor, and the type of window used in the forward FFT. The inverse FFT always uses a rectangular window. This algorithm recomputes the threshold for each successive block of data and so adapts to changing background levels.

The user has the option of storing the spectra before and after excision. The user specifies a filename stem, and the Vulnerability Metric appends ".in" and ".out" to stem to form the names of the files storing the spectra before and after excision respectively. The Vulnerability Metric writes all signal files to the directory /spwdata/vmsigs.

Taking the FFT delays the signal by a number of iterations equal to the FFT size, so the signal value presented at the DELAY OUT port is equal to the value at the DELAY IN port plus the FFT size.

MENU PATH

Communicator > Preprocessor > Adaptive Threshold Excision

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

OUT	Complex

PARAMETERS

exceedfactor	Exceedance factor
nexempt	Number of points excluded in the calculation of the mean square magnitude
blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT

COMMUNICATOR'S ADAPTIVE THRESHOLD EXCISION SYSTEM, continued

store_spectra	Store spectra?
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exadapt

COMMUNICATOR'S FIXED EXCISION SYSTEM

DESCRIPTION

Performs a spectral analysis of the input and removes the strongest components. The user specifies the size of the FFT's, the number of strongest bins to be excised, and the type of window used in the forward FFT. The inverse FFT always used a rectangular window. The frequency resolution, or bin size, is the sampling frequency divided by the FFT size. If the FFT size is N, simulation samples numbered 0 through N-2 are in the first block, N-1 through N-2 in the second block, and so on.

The user has the option of storing the spectra before and after excision. The user specifies a filename stem, and the Vulnerability Metric appends ".in" and ".out" to stem to form the names of the files storing the spectra before and after excision respectively. The Vulnerability Metric writes all signal files to the directory /spwdata/vmsigs.

Taking the FFT delays the signal by a number of iterations equal to the FFT size, so the signal value presented at the DELAY OUT port is equal to the value at the DELAY IN port plus the FFT size.

MENU PATH

Communicator > Preprocessor > Fixed Excision

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

OUT	Complex

PARAMETERS

nlargest	Number of bins to be excised
blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
store_spectra	Store spectra?
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/enfixed

COMMUNICATOR'S FREQUENCY BASED EXCISION

DESCRIPTION

Optionally applies one of three frequency based excision techniques: fixed, adaptive threshold, or whitening.

MENU PATH

Communicator > Preprocessor > Composite Freq Based Excision

INPUT PORTS

IN	Complex
DELAY_IN	Real

OUTPUT PORTS

OUT	Complex
DELAY_OUT	Real

PARAMETERS

index	Selection index (0-no excision, 1-Fixed, 2-Adaptive, 3-Whitening)
store_spectra	Store spectra? (index> 0)
blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
nlargest	Number of bins to be excised
exceedfactor	Exceedance factor)index = 1)
nexempt	Number of points excluded in the calculation of mean square magnitude (index = 2)
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/freq_excision

COMMUNICATOR'S WHITENING EXCISION SYSTEM

DESCRIPTION

Performs a spectral analysis of the input and divides each spectral component by its magnitude resulting in flat output spectrum.. The user specifies the size of the FFT's, and the type of window used in the forward FFT. The inverse FFT always uses a rectangular window. The spectral resolution, or bin size, is the sampling frequency divided by the FFT size. If the input spectrum in any bin is smaller than the machine's arithmetic resolution, the magnitude is set to 1, and the phase is arbitrarily set to 0. If the FFT size is N, simulation samples numbered 0 through N-2 are in the first block, N-1 through N-2 in the second block, and so on.

The user has the option of storing the spectra before and after excision. The user specifies a filename stem, and the Vulnerability Metric appends ".in" and ".out" to stem to form the names of the files storing the spectra before and after excision respectively. The Vulnerability Metric writes all signal files to the directory /spwdata/vmsigs.

Taking the FFT delays the signal by a number of iterations equal to the FFT size, so the signal value presented at the DELAY OUT port is equal to the value at the DELAY IN port plus the FFT size.

MENU PATH

Communicator > Preprocessor > Whitening Excision

INPUT PORTS

IN	Complex
DELAY_IN	Real
HOLD	Real
DELAY_OUT	Real

OUTPUT PORTS

OUT	Complex

PARAMETERS

blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT

store_spectra Store spectra?

sfreq Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exwhite

COMPLEX INPUT SWITCH

DESCRIPTION

Selects the output from one of the complex inputs. Parameters specify both the number of permitted inputs, and which one is used for the output. The module tests the value of the both parameters, and terminates with an error message if either is out of range. Setting the parent module parameter permits the error message to give the containing module of the instance that has an error.

MENU PATH

Utilities > Logic > Switch, Complex, Parameter

INPUT PORTS

IN_0	Complex
IN_1	Complex
IN_2	Complex
IN_3	Complex
IN_4	Complex
IN_5	Complex

OUTPUT PORTS

OUT	Complex

PARAMETERS

no_of_inputs	Number of inputs (1-6)
index	Selection index (0-no_of_inputs-1)

LIBRARY / FUNCTION

utlib/switch_in

Coded in C

COMPLEX MOVING AVERAGE

DESCRIPTION

Outputs the input averaged over an immediately preceding time interval.

MENU PATH

Utilities > Estimators > Complex Moving Avge

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

sfreq	Simulation sampling frequency
inttime	Length if signal averaging time interval

LIBRARY / FUNCTION

utlib/cmplx_mv_avge

COMPLEX SWITCHED INPUT

DESCRIPTION

Outputs the top signal input if the control parameter is 0 and outputs the bottom signal input if the control parameter is 1.

MENU PATH

Utilities > Logic > Switch, Complex, Signal

INPUT PORTS

IN	Complex
CONTROL	Integer
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

None

LIBRARY / FUNCTION

utlib/cmplx_sw_in

COMPLEX VARIANCE

DESCRIPTION

Outputs the mean and variance of the input signal, computed from the beginning of the simulation.

MENU PATH

Utilities > Estimators > Complex Variance

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

MEAN	Complex
VAR	Real

PARAMETERS

None

LIBRARY / FUNCTION

utlib/cmplx_var

CORRELATION DOUBLER SYSTEM

DESCRIPTION

Uses a communication signal input and a pair of background inputs to form a signal present pair and signal absent pair, and passes the signal present pair through one correlation doubler detector and the signal absent pair through a second correlation doubler detector. The result of processing the signal present is the H_1 output and the result of processing the signal absent pair is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Freq Multr > Corr Doubler, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND1	Complex
BACKGROUND2	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
collect_time	Output integration time in detector systems
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Simulation sampling frequency
nsamples	Number of samples in simulation
fda	Data rate

LIBRARY / FUNCTION

delib/cor_doub

CORRELATION DOUBLER CORE

DESCRIPTION

Simulates the action of a correlation doubler on complex envelope inputs. Provides optional storage of the outputs of one receiver, the multiplier and the detector system.

MENU PATH

Interceptor > Detector > Freq Multr > Corr Doubler, core

INPUT PORTS

IN1	Complex
IN2	Complex
HOLD	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

write_sigs	1 to store intermediate signals, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
int_time	Integration time in output integrator
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Numbers of samples in simulation

LIBRARY/FUNCTION

delib/cor_doub_core

CORRELATION RADIOMETER SYSTEM

DESCRIPTION

Uses a communication signal input and a pair of background inputs to form a signal present pair and signal absent pair, and passes the signal present pair through one correlation radiometer detector and the signal absent pair through a second correlation radiometer detector. The result of processing the signal present pair is the H_1 output and the result of processing the signal absent pair is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Radiometer > Correlation, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND1	Complex
BACKGROUND2	Complex

OUTPUT PORTS

H1	Complex
H2	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
collect_time	Output integration time in detector frequency
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for non weak signal, simulated background mode, o otherwise
fces	Signal center frequency
sfreq	Sampling frequency

nsamples	Number of samples in simulation
fda	Data rate

USAGE EXAMPLE

SSVM>Demonstrates>Correlation Radiometer

LIBRARY / FUNCTION

delib/cor_radiom

CORRELATION RADIOMETER CORE

DESCRIPTION

Simulates the action of a correlation doubler on complex envelope inputs. Provides optional storage of the outputs of one receiver, the multiplier and the detector system.

MENU PATH

Interceptor > Detector > Freq Multr > Corr Doubler, core

INPUT PORTS

IN1	Complex
IN2	Complex
HOLD	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

write_sigs	1 to store intermediate signals, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
int_time	Integration time in output integrator
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation

USAGE EXAMPLE

SSVM>Demonstrations>Correlation Radiometer

Double click on "CORRELATION RADIOMETER" module

LIBRARY / FUNCTION

delib/cor_radiom_core

CUSTOM INTERFERENCE

DESCRIPTION

Reads an interference configuration supplied by the user and adds the result to the signal. The user specifies the number of interferers, the configuration center, interferer powers, in dB above communication signal power, and the widths. The bandwidth is implemented by BPSK modulating each interference carrier frequency equal to the bandwidth. The user specifies the name of the file configuration, the default name begins custom.conf. The format of the file is a format created by the Random Interference Module, so that configuration randomly and saved may be reused directly by the Custom Interference Module.

The mathematical form of the interference is the same as that detailed for Random Interference.

MENU PATH

Background > Interference > Custom

INPUT PORT

None

OUTPUT PORT

OUT Complex

INPUT FILE

data/custom.conf

The input file has the following format:

Interference Configuration

Num_Freqs: 100 Center_Freq: 1.000000e+06
-3.363100e+05 +2.273510e+05 ... +3.575460e+05

Powers:
9.535084e+00 9.055280e+00 ... 1.087717e+01

Widths:
3.341063e+02 4.591780e+02 ... 8.631209e+02

PARAMETERS

conf_file	Configuration input filename, defaults to custom.conf
simtime	Simulation time
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

bklib/ifcus

Coded in C

DATA SOURCE

DESCRIPTION

Outputs a pseudorandom binary sequence.

MENU PATH

Utilities > Input /Output > Data Source

INPUT PORTS

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

fda	Data rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/data_source

DEINTERLEAVER

DESCRIPTION

Implements a block deinterleaver, which reverses the effect of the block interleaver. An NxM deinterleaver is the same as an MxN interleaver.

MENU PATH

Communicator > Demodulator > Deinterleaver

INPUT PORTS

IN1	Real
HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

ncols	Number of columns in corresponding interleaver matrix
nrows	Number of rows in corresponding interleaver matrix
fda	Data rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

modlib/deinterleave

DELAY CHIP RATE DETECTOR SYSTEM

DESCRIPTION

Transmits the signal present and signal absent inputs to separate delay chip rate detectors. Outputs the results, as well as the noise bandwidth.

MENU PATH

Interceptor > Detector > Chip Rate > Delay, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORT

H1	Complex
H0	Complex
NOISE_BW	Real

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
delay_factor	Delay, in units of half the estimated chip duration
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	Data rate

collect_time Output integration time in detector systems

LIBRARY / FUNCTION

delib/delay_chip

DELAY CHIP RATE DETECTOR CORE

DESCRIPTION

Simulates the action of a delay chip rate detector with complex envelope input. Provides optional storage of intermediate signals.

MENU PATH

Interceptor > Detector > Chip Rate > Delay, core

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

write_sigs	1 to store intermediate signals, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
collect_time	Output integration time
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/delay_chip_cor

DELAY, FFT SEARCH, CHIP RATE DETECTOR

DESCRIPTION

Implements a delay chip rate detector with an FFT output stage to permit detection at multiple chip rates. In addition the output stage provides filtering, subsampling and block averaging. The user can therefore tailor the search to cover efficiently the range of chip rate uncertainty. The documentation of the Envelope Spectrum Detector provides detailed information on the filtering, subsampling and block averaging.

MENU PATH

Interceptor > Detector > Chip Rate > Delay, FFT Search

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

None

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
store_signals	1 to store intermediate signals, 0 otherwise
sfreq	Sampling frequency
fces	Signal center frequency
fchd	Interceptor chip rate estimate
delay_factor	Delay
start_time	Time to begin collecting date to determine averaged spectrum

spec_bwif	Filter width (lowpass, 2-sided)
subsampling	Subsampling order
fft_size	FFT size, power of 2
window_type	Window (blackman, bartlett, hamming, hanning, rectangular)
spec_avg_block	Number of spectra to average
freq_zoom	Frequency zoom, power of 2

USAGE EXAMPLE

SSVM>Demonstrations>Search Chip Rate Detector

LIBRARY / FUNCTION

delib/delay_chip_fft

DELAY, STEPPING SEARCH, CHIP RATE DETECTOR

DESCRIPTION

Combines a delay chip rate detector with a two-dimensional search that scans receiver center frequency and chip rate. The chip rate cycles through a fixed number of values specified by the user, while the receiver center frequency keeps incrementing throughout the simulation, once after each cycle through the chip rate, so the number of center frequencies used depends on the number of collects simulated and the number of chip rates used. The user controls the order and passband edge attenuation of the receiver filter and specifies the name, **`filename`**, of the file to which the results are written.

After the simulation, the user obtains a Gnuplot screen plot by entering

```
DelSrch  filename
```

in an operating system window. The plot also displays the name of the underlying data file, To obtain a hard copy in addition to the screen plot, the user enters

```
DelSrch  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`DelSrch` is a Bourne shell script which selects and edits the file **`DelSrch.hard.template`** if the flag **`-h`** is present, and **`DelSrch.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`DelSrch.gnu`**. The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`DelSrch`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Detector > Chip Rate > Delay, Step Search

INPUT PORTS

IN	Complex
HOLD	Complex

OUTPUT PORTS

None

DELAY, STEPPING SEARCH, CHIP RATE DETECTOR, continued

PARAMETERS

start_fced	Starting receiver center frequency
step_fced	Receiver center frequency increment
bwif	Receiver bandwidth
fchd	Interceptor chip rate estimate
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
store_signals	1 to store intermediate signals, 0 otherwise
start_fchd	Starting chip rate
step_fchd	Chip rate increment
collect_time	Output integration time
sfreq	Sampling frequency
fces	Signal center frequency

USAGE EXAMPLE

SSVM>Demonstrations>Search Chip Rate Detector

LIBRARY / FUNCTION

delib/del_chip_srch

DENSITY BASED PREPROCESSOR

DESCRIPTION

Implements the Digital Density Detector. Estimates the probability density of the additive noise by computing a histogram over a fixed data block. The probability density estimate is used to compute a nonlinearity which is then applied to the input signal, to produce a nonlinearly processed data signal, delayed by the length of the block. The vector corresponding to the accumulated histogram values is also output, on a block-by-block basis.

MENU PATH

None

INPUT PORT

IN	Complex

OUTPUT PORTS

SIG	Complex
HIST	Real Vector

PARAMETERS

blk_size	Processing block size
maxval	Clipping Level
nbins	Number of bins for histogram computation

LIBRARY / FUNCTION

prlib/ddd_sys

DETECTOR INPUT SWITCH

DESCRIPTION

Switches the signal present, H_1, and signal absent, H_0, detector inputs to the two detector cores, as required by the selected Operating Mode. The Mode selection is done for the user based on the specification of global parameters in the Vulnerability Metric Viewpoint.

MENU PATH

Interceptor > Detector > Utilities > Detector Input Switch

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Real

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

long_stat_coll	Long, stationary collect?
quad_weak	Quadratic processing, with weak signal?
Bideal_bkgd	Ideal background?

LIBRARY / FUNCTION

delib/mode_switch

DIGITAL TO ANALOG CONVERTER

DESCRIPTION

Converts a binary vector to a real number. The low order bit is interpreted as the one's coefficient, the next higher bit as the two's coefficient, and so on. The output is a real number between 0 and 2 (number of bits in) -1.

MENU PATH

Utilities > Signal Processing > D/A Converter

INPUT PORTS

IN	Real Vector
LOAD_IN	Real
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

IN_IOVEC_LEN	Number of bits in the input vector
write_sigs	1 to store intermediate signals, 0 otherwise
fda	Data rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/d_a

Coded in C

DIRECT SEQUENCE DESPREADER

DESCRIPTION

Reverses the effect of direct sequence spreading. Since the spreading sequence is conveyed by means of a global connector, a single block serves from arbitrary number of phases, and for any type of jittering of agility.

MENU PATH

Spread Spectrum/DS Despreading

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

None

LIBRARY / FUNCTION

walib/ds_despreader

ENVELOPE CHIP RATE DETECTOR SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form a signal present input to one detector and a signal absent input to another detector. The detectors are each envelope chip rate detectors. The result of processing the signal present case is the H_1 output and the result of processing the signal absent case is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Chip Rate > Envelope, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORT

H1	Complex
H2	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
fchd	Interceptor's estimate of chip rate
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	data rate
collect_time	Output integration time in detector systems

LIBRARY / FUNCTION

delib/envel_chip

ENVELOPE CHIP RATE DETECTOR CORE

DESCRIPTION

Simulates the action of an envelope chip rate detector with complex envelope input. Provides optional storage of intermediate signals.

MENU PATH

Detectors > Chip Rate > Envelope, core

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

write_signals	1 to store intermediate signal, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
collect_time	Output integration time
sfreq	Sampling frequency
fces	Signal center frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/envel_chip_cor

ENVELOPE, STEPPING SEARCH CHIP RATE DETECTOR

DESCRIPTION

Combines a envelope chip rate detector with a two-dimensional search that scans receiver center frequency and chip rate. The chip rate cycles through a fixed number of values specified by the user, while the receiver center frequency keeps incrementing throughout the simulation, once after each cycle through the chip rate, so the number of center frequencies used depends on the number of collects simulated and the number of chip rates used. The user controls the order and passband edge attenuation of the receiver filter and specifies the name,**`filename`**, of the file to which the results are written.

After the simulation, the user obtains a Gnuplot screen plot by entering

`EnvSrch  filename`

in an operating system window. The plot also displays the name of the underlying data file, To obtain a hard copy in addition to the screen plot, the user enters

`EnvSrch  -h  filename`

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv  PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`EnvSrch` is a Bourne shell script which selects and edits the file **`EnvSrch.hard.template`** if the flag **`-h`** is present, and **`EnvSrch.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`EnvSrch.gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`EnvSrch`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Detector > Chip Rate > Envelope, Step Search

INPUT PORTS

IN	Complex
HOLD	Complex

OUTPUT PORTS

None

ENVELOPE, STEPPING SEARCH, CHIP RATE DETECTOR, continued

PARAMETERS

start_fced	Starting receiver center frequency
step_fced	Receiver center frequency increment
bwif	Receiver bandwidth
fchd	Interceptor chip rate estimate
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
store_signals	1 to store intermediate signals, 0 otherwise
start_fchd	Starting chip rate
step_fchd	Chip rate increment
no_of_chip_rates	Number of chip rates Chip rate increment
collect_time	Output integration time
sfreq	Sampling frequency
fces	Signal center frequency

USAGE EXAMPLE

SSVM>Demonstrations>Search Chip Rate Detector

LIBRARY / FUNCTION

delib/envl_chip_srch

ENVELOPE SPECTRUM

DESCRIPTION

Lowpass filters the incoming complex signal; buffers the signal into blocks;, subsamples the blocks; forms a spectrum by performing an FFT and magnitude squaring the output; averages the results over a number of blocks, and stores the logarithm of the results for subsequent plotting.

After the simulation, the user enters

```
Spectrum  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user in the spectrum module. The result is a Gnuplot screen plot showing spectral strength as a function of frequency. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

```
Spectrum  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`Spectrum` is a Bourne shell script which selects and edits the file **`Spectrum.hard.template`** if the flag **`-h`** is present, and **`Spectrum.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`Spectrum.gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Spectrum`** can be invoked from any directory without specifying a path.

MENU PATH

None

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

None

ENVELOPE SPECTRUM, continued

PARAMETERS

start_time	Starting receiver center frequency
bwif	Receiver bandwidth
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
subsampling	Subsampling order
fft_size	FFT size, power of 2
window_type	Window (blackman, bartlett, hamming, hanning, rectangular)
spec_avg_block	Number of spectra to average
center_freq	Center frequency
freq_zoom	Frequency zoom, power of 2
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

delib/env_spectrum

ERROR HANDLER

DESCRIPTION

Produces an error message in a pop-up window (whenever IN > 0.0)

MENU PATH

Utilities > Logic > Error Handler

INPUT PORT

IN	Real

OUTPUT PORT

None

PARAMETERS

parent_module	Parent module string
message	Error message string

LIBRARY / FUNCTION

utlib/error_handler

Coded in C

EXCEEDANCE

DESCRIPTION

Treats the inputs as the outputs of the histogram module, that is as the underflow, binned, and overflow counts of a real signal, and computes the exceedance. The exceedance of each bin is the number of counts larger than that bin's lower boundary, that is the number of counts in that bin plus the number of counts in all higher bins. The overflow is treated as a bin so the first, or highest component in the output vector is equal to the OVER input. The UNDER input is passed to the output for subsequent processing.

MENU PATH

Interceptor> Metrics > Exceedance

INPUT PORTS

OVER	Real
BINS	Real Vector
UNDER_in	Real
HOLD	Real

OUTPUT PORTS

OUT	Real vector, of dimension equal to one plus the dimension of the BINS input
UNDER_out	Real

PARAMETERS

BINS_IOVEC_LEN

Number of components in the input vector BINS

LIBRARY / FUNCTION

vulib/exceedance

Coded in C

ENVELOPE, FFT SEARCH CHIP RATE DETECTOR

DESCRIPTION

Implements an envelope chip rate detector with an FFT output stage to permit detection at multiple chip rates. In addition the output stage provides filtering, subsampling and block averaging. The user can therefore tailor the search to cover efficiently the range of chip rate uncertainty. The documentation of the Envelope Spectrum Detector provides detailed information on the filtering, subsampling and block averaging.

MENU PATH

Interceptor > Detectors > Chip Rate > Envelope, FFT Search

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

None

PARAMETERS

write_signals	1 to store intermediate signal, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
store_signals	1 to store intermediate signals, 0 otherwise
sfreq	Sampling frequency
fces	Signal center frequency
start_time	Time to begin collecting data to determine averaged spectrum
spec_bwif	Filter width (lowpass, 2-sided)
subsampling	Subsampling order

ENVELOPE, FFT SEARCH CHIP RATE DETECTOR, continued

fft_size	FFT size, power of 2
window_type	Window (blackman, bartlett, hamming, hanning, rectangular)
spec_avg_block	Number of spectra to average
freq_zoom	Frequency zoom, power of 2

USAGE EXAMPLE

SSVM>Demonstrations>Search Chip Rate Detector

LIBRARY / FUNCTION

delib/env_chip_fft

FILE STEPPER

DESCRIPTION

Produces a stepped output which cycles through a set of values. The user stores arbitrary values in a file, and specifies a scaling factor and an offset. The user also controls the dwell time, which is the time interval between changes in the output.

MENU PATH

Utilities > Sources > File Stepper

INPUT

HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

dwell_time	Time interval between changes in output
filename	Name of file used for file mode
scale_factor	Scaling applied to values from file
offset	Offset applied to scaled values from file
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/file_stepper

FILTERING

DESCRIPTION

Filters the complex envelope input with a lowpass filter that may be offset from the signal center frequency. The user controls the filter offset and bandwidth. The filter is a Butterworth IIR filter with an adjustable order preset at 20 and an adjustable passband edge attenuation preset to 3dB. The filtering module may only be used in the two weak signal operational modes of the Vulnerability Metric.

Communicators adjust the signal power so the transmitted power is constant, independent of the filtering. Direct simulation of power compensation would require an additional simulation to measure the impact of filtering on power. To maintain computational efficiency, the Vulnerability Metric allows the filter module to reduce the transmitted power, but measures and stores the reduction factor on disk at the end of the simulation. In both of the weak signal modes, the vulnerability assessment at the end of the simulation reads the power reduction factor and rescales the signal strength. From the user's perspective, the vulnerability evaluation runs exactly as if transmitted power were maintained constant.

MENU PATH

Spread Spectrum > Filtering

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

offset	Offset of filter passband center form signal center frequency
bw	Filter bandwidth
filter_order	Receiver filter order
edge_atten	Receiver passband edge attenuation, dB power
sfreq	Sampling frequency
fces	Signal center frequency
nsamples	Number of iterations in simulation

LIBRARY / FUNCTION

walib/filter

FIXED EXCISION CORE

DESCRIPTION

Performs a spectral analysis of the input and removes the strongest components. The user specifies the length of the vectors, and the number of strongest bins to be excised.

MENU PATH

None

INPUT PORTS

X-re	Real Vector
X-im	Real Vector

OUTPUT PORTS

Y-re	Real Vector
Y-im	Real Vector

PARAMETERS

blocksize	Number of points used in the FFTs
nlargest	Number of bins to be excised
window_type	Type of window used in the forward FFT
sfreq	Sampling frequency

LIBRARY / FUNCTION

prlib/exfixed_cmp

Coded in C

FREQUENCY AGILITY

DESCRIPTION

Shifts the incoming signal's frequency and phase by random offsets which remain constant over time intervals called hops, and vary pseudorandomly between hops. Hop length, generated by the AGILITY module, also vary pseudorandomly, and are rounded to the nearest multiple of the data symbol duration. Typically the difference between the minimum and maximum hop lengths exceeds the data symbol duration, so the FREQUENCY AGILITY module performs slow frequency hopping with fluctuation hop lengths.

The user specifies the minimum and maximum block lengths; the number of frequency offsets; the maximum one sided frequency deviation; and whether the block lengths and block boundary pulses are stored. If the input signal has constant unit magnitude, the output power is 1.

MENU PATH

Spread Spectrum > Frequency Agility

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

min_duration	Minimum block length
max_duration	Maximum block length
nfreq	Number of available frequency offsets
maxfreqs	Maximum frequency deviation
store_hops	1 to store block lengths and block boundary pulses, 0 otherwise
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

walib/freq_agility

FREQUENCY DOUBLER SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form a signal present input to one detector and a signal absent input to another detector. The detectors are each frequency doubler detectors. The result of processing the signal present case is the H_1 output and the result of processing the signal absent case is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Freq Multr > Doubler, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORT

H1	Complex
H0	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver bandwidth
fchd	Interceptor's estimate of chip rate
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	data rate
collect_time	Output integration time in detector systems

LIBRARY / FUNCTION

delib/doubler

FREQUENCY DOUBLER CORE

DESCRIPTION

Simulates the action of an frequency doubler detector with complex envelope input. Provides optional storage of intermediate signals.

MENU PATH

Detectors > Freq Multrs > Doubler, core

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

write_signals	1 to store intermediate signal, 0 otherwise
fced	Receiver center frequency
bwif	Receiver bandwidth
collect_time	Output integration time
sfreq	Sampling frequency
fces	Signal center frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/doubler_core

FREQUENCY HOPPING

DESCRIPTION

Shifts the incoming signal's frequency and phase by random offsets. The offsets are constant during intervals defined as hops, and are reselected for each hop. If the input signal has constant unit magnitude, the output power is 1.

MENU PATH

Spread Spectrum > Frequency Hopping

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

hop_rate	Hop rate
nfreq	Number of available frequency offsets
maxfreqs	Maximum frequency deviation
write_freqs	1 to store frequency offsets
sfreq	Sampling frequency

LIBRARY / FUNCTION

walib/incoh_freq_hop

FREQUENCY HOPPING DOWNCONVERTER

DESCRIPTION

Coherently reverses the effect of the Frequency Hopping and Frequency Agility modules.

MENU PATH

Communicator/Preprocessor > Freq Hop Downconvert

INPUT PORTS

IN	Complex
DELAY_IN	Real
HOLD	Real

OUTPUT PORTS

OUT	Complex
DELAY_OUT	Real

PARAMETERS

sfreq	Sampling frequency

LIBRARY / FUNCTION

prlib/fh_downconvert

FSK DEMODULATOR

DESCRIPTION

A demodulator of M-ary frequency shift keying.

MENU PATH

Communicator > Demodulator > FSK

INPUT PORT

IN	Complex
DELAY_IN	Real

OUTPUT PORT

OUT	Real
DELAY_OUT	Real

PARAMETERS

m	Alphabet size
fsep	Tone separation
fda	Bit rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

modlib/fsk_demod

FSK MODULATOR

DESCRIPTION

A M-ary frequency shift keyed modulator. Either coherent or incoherent modulation may be chosen.

MENU PATH

Communicator > Modulator > FSK

INPUT PORT

IN	Real
DELAY_IN	Real

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

m	Alphabet size
fsep	Tone separation
coherence_flag	Flag to determine whether coherent FSK is chosen. 1 of coherent, 0 for incoherent.
fda	Bit rate
sfreq	Sampling frequency

LIBRARY / FUNCTION

modlib/fsk_mod

FSK SOURCE

DESCRIPTION

Outputs a narrowband complex envelope resulting from interleaving a pseudorandom bit source, and using the result as the data stream for a frequency shift keying modulation. The user sets the numbers of columns and rows to define the interleaving, and accepting the default value of 1 for both produces a non-interleaved data stream. The user also sets the alphabet size, that is the number of frequencies to be used in the modulation; the frequency separation, and whether the modulation is coherent or not.

MENU PATH

Communicator > Modulated Source > FSK

INPUT PORT

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

ncols	Interleaver columns
nrows	Interleaver rows
m	Alphabet size
fsep	Tone Separation
coherence_flag	Flag to determine whether coherent FSK is chosen, 1 for coherent, 0 for incoherent.
fda	Bit rate
sfreq	Sampling rate

USAGE EXAMPLES

SSVM>Demonstrations> Correlation Radiometer
SSVM>Demonstrations> Scanner

LIBRARY / FUNCTION

modulib/fsk_source

ESTIMATED OUTPUT SNR

DESCRIPTION

Estimates the output signal to noise ratio

$$(S/N)_0 = \frac{\left| E\{z\,|H_1\} - E\{z\,|H_0\} \right|^2}{Var\{z\,|H_1\}}$$

The denominator is a variance, and the first output of the variance estimators is zero, so the module's first output sample should be ignored.

MENU PATH

Interceptor > Metrics > Estimated Output SNR

INPUT PORTS

H1	Complex
H0	Complex
HOLD	Real

OUTPUT PORTS

OUT	Complex

PARAMETERS

None

LIBRARY / FUNCTION

vulib/full_snro

GAMMA NOISE

DESCRIPTION

Produces a discrete complex envelope model of continuous bandpass white noise whose magnitude is distributed as a Gamma random variable, and whose phase is uniformly distributed. The marginal distributions of the real and imaginary components are not expressible as elementary functions, but decay exponentially. The probability density of the magnitude is given by

$$xe^{-x}$$

The noise power in both the Gaussian and Gamma Noise modules is parametrized by the communicator's input signal to noise ratio, which is computed for the case where only one noise module is present. Using more than one noise module at a time requires manually determining the input signal to noise ratio.

MENU PATH

Background > Noise > Gamma

INPUT PORT

IN	None

OUTPUT PORTS

OUT	Complex

PARAMETERS

dB_in	Input signal to noise ratio, for the case where no other noise module is in use
sfreq	Sampling frequency
fda	Data rate
nsamples	Number of samples in the simulation

LIBRARY / FUNCTION

bklib/whi_gam_nse

HETERODYNE

DESCRIPTION

Multiplies the input by a complex tine.

MENU PATH

Utilities > Signal; Processing > Heterodyne

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

het_freq	Heterodyne frequency
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/heterodyne

HF INTERFERENCE

DESCRIPTION

Adds to the signal an interference background generated from a stored configuration chosen to be characteristic of typical HF operating environments. The user selects the center and width of the operating band, and the module generates the interferers which lie in the selected band.

The mathematical form of the interference the same as that detailed for Random Interference.

MENU PATH

Background > Interference > HF

INPUT PORTS

None

OUTPUT PORT

OUT	Complex

PARAMETERS

if_cnet_freq	Configuration center frequency
if_bandwidth	Configuration one-sided bandwidth
simtime	Simulation time
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

bklib/ifhf

Coded in C

HISTOGRAM

DESCRIPTION

Accumulates a histogram of the incoming signal. The user specifies the lowest and highest values captured and the number of bins into which the intervening interval is divided. The module counts the number of input samples falling into each bin, as well as the number below the lowest captured value and the number above the highest captured value.

MENU PATH

Interceptor > Metrics > Histogram

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

OVER	Real
OUT	Real vector, of dimension equal to the number of bins
UNDER	Real

PARAMETERS

min	Lowest value captured
max	Highest value captured
DEFAULT_VECLEN	Number of bins

LIBRARY / FUNCTION

vulib/histogram

Coded in C

HOLD SWITCH

DESCRIPTION

If the selection index parameter is zero, sets all outputs to TRUE. Otherwise sets all outputs to TRUE except one, which is identified by a selection index parameter. The module sets the selected port to FALSE. Another parameter specifies the number of options, that is the number of outputs available for the selection, plus one for the case where none is selected. The module sets the value of the both parameters, and terminates with an error message if either is out of range, Setting the parent module parameter permits the error message to give the containing module of the instance that has an error.

MENU PATH

Utilities > Logic > Hold Switch

INPUT PORT

None

OUTPUT PORT

OUT_1	Real
OUT_2	Real
OUT_3	Real
OUT_4	Real
OUT_5	Real

PARAMETERS

no_of_options	Number of options (1-6)
index	Selection index, between 0 and (number of options - 1)

LIBRARY / FUNCTION

utlib/switch_hold

INCOMPLETE RUN

DESCRIPTION

Outputs TRUE until the number of executed iterations exceeds the total number of iterations in the simulation minus 2, when the output becomes False.

MENU PATH

Utilities > Incomplete Run

INPUT PORT

HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

sfreq	Simulation sampling frequency
collect_time	Detector collect time
Collects	Number of collects

LIBRARY / FUNCTION

utlib/incomplete_run

INTERCEPT COUPLER

DESCRIPTION

Uses the transmitted signal and background noise to form the signal plus noise (H1) and noise alone (H0) hypothetical waveforms for input to the intercept detectors.

MENU PATH

Channel > Couplers > Interceptor

INPUT PORT

SIGNAL	Real
BACKGROUND	Real

OUTPUT PORT

H1	Real
H0	Real

PARAMETERS

None

USAGE EXAMPLES

SSVM>Demonstrations>Search Chip Rate Detector
SSVM>Demonstrations>Scanner

LIBRARY / FUNCTION

chlib/intcpt_coupler

INTERCEPTOR'S ADAPTIVE THRESHOLD EXCISION SYSTEM

DESCRIPTION

Applies adaptive threshold excision to both the signal present H_1 and signal absent H_0 waveforms.

MENU PATH

Interceptor > Preprocessor > Adaptive Threshold Excision

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

exceedfactor	Exceedance factor
nexempt	Number of points excluded in the calculation of mean square magnitude
blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exadapt_int

INTERCEPTOR'S FIXED EXCISION SYSTEM

DESCRIPTION

Applies fixed excision to both the signal present H_1 and signal absent H_0 waveforms.

MENU PATH

Interceptor > Preprocessor > Fixed Excision

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

nlargest	Number of bins to be excised
blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exfixed_int

INTERCEPTOR'S WHITENING EXCISION SYSTEM

DESCRIPTION

Applies whitening excision to both the signal present H_1 and signal absent H_0 waveforms.

MENU PATH

Interceptor > Preprocessor > Whitening Excision

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/exwhite_int

INTERCEPTOR'S COMPOSITE PREPROCESSOR

DESCRIPTION

Optionally applies one of three frequency based excision techniques: fixed adaptive threshold, or whitening.

MENU PATH

Interceptor > Preprocessor > Composite

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

index	Selection index (0-no excision, 1- Fixed, 2-Adaptive, 3-Whitening)
sfreq	Simulation sampling frequency

Additional parameters depending in **index**: see "INTERCEPT FIXED EXCISION", "INTERCEPT ADAPTIVE EXCISION", and "INTERCEPT WHITENING EXCISION"

LIBRARY / FUNCTION

prlib/intercept

INTERLEAVER

DESCRIPTION

Implements a block interleaver, which permutes data by sequentially loading an NxM matrix with input bits row by row, and reading the bits out column by column.

MENU PATH

Communicator > Modulator > Interleaver

INPUT PORTS

IN	Real
HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

ncols	Number of columns in interleaver matrix
nrows	Number of rows in interleaver matrix
fda	Data rate
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

modlib/interleaver

TRANSPOSE

DESCRIPTION

Implements the core a block interleaver, permuting data by sequentially loading an NxM matrix with input bits row by row, and reading the bits out column by column. This inner block is called by both the top level interleaver and deinterleaver.

MENU PATH

None

INPUT PORTS

IN	Real
HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

ncols	Number of columns in interleaver matrix
nrows	Number of rows in interleaver matrix
initial_value	Initial value in interleaver matrices

LIBRARY / FUNCTION

modlib/transpose

JITTER

DESCRIPTION

Produces trigger pulses at randomly occurring time steps. The random timing of the zeros is constrained: referring to the interval between successive zeros as a chip, there are always an integer number of chips in a data symbol, and the chip rate only changes at data symbol boundaries. The instantaneous chip rate is generated randomly by truncating a continuous uniform random variable to the nearest integer multiple of the data rate. The user specifies the mean and maximum deviation of the continuous random variable, with the mean parametrized as a multiple of the data rate, and the size of the chip rate fluctuations parametrized as a percentage of the average rate. The user also controls the interval between chip rate changes, parametrized as a multiple of the data symbol length, and the user may write to disk the instantaneous chip rate and the module output.

The terms "jitter" and "dither" are synonymous.

MENU PATH

Spread Spectrum > Jitter

INPUT PORT

None

OUTPUT PORT

OUT	Real

PARAMETERS

nchav	Average number of chips per symbol
nsybj	Number of symbols between chip rate changes
hw	Maximum allowed deviation of chip rate from average chip rate
write_jitter	1 to write instantaneous chip rate and chip release impulses to files, 0 otherwise
fda	Data rate
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

walib/jitter

JITTERED 8ary DS SPREADER

DESCRIPTION

Multiplies the complex input by a 8-ary complex pseudonoise sequence with a fluctuation chip rate. The user specifies the average chip rate as a multiple of the data rate, the size of the chip rate fluctuations as a percentage, and the interval between chip rate changes as a multiple of the data symbol length. Optionally stores the instantaneous chip rate and the sequence of impulses that trigger each chip. If the input signal has constant unit magnitude, the output power is 1.

MENU PATH

Spread Spectrum > Jittered Direct Sequence > 8ary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nchav	Average number of chips per symbol
nsybj	Number of symbols between chip rate changes
hw	Maximum allowed deviation of chip rate from average chip rate
write_jitter	1 to write instantaneous chip rate and chip release impulses to files, 0 otherwise
fda	Data rate
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

walib/j_8ary_ds

JITTERED BINARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a binary complex pseudonoise sequence with a fluctuation chip rate. The user specifies the average chip rate as a multiple of the data rate, the size of the chip rate fluctuations as a percentage, and the interval between chip rate changes as a multiple of the data symbol length. Optionally stores the instantaneous chip rate and the sequence of impulses that trigger each chip. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization.

Outputs the spreading sequence as a global output connector called ‘code’.

MENU PATH

Spread Spectrum > Jittered Direct Sequence > Binary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nchav	Average number of chips per symbol
nsybj	Number of symbols between chip rate changes
hw	Maximum allowed deviation of chip rate from average chip rate
write_jitter	1 to write instantaneous chip rate and chip release impulses to files, 0 otherwise
fda	Data rate
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

walib/j_binary_ds

JITTERED QUATERNARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a quaternary complex pseudonoise sequence with a fluctuation chip rate. The user specifies the average chip rate as a multiple of the data rate, the size of the chip rate fluctuations as a percentage, and the interval between chip rate changes as a multiple of the data symbol length. Optionally stores the instantaneous chip rate and the sequence of impulses that trigger each chip. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization.

Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Jittered Direct Sequence > Quaternary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nchav	Average number of chips per symbol
nsybj	Number of symbols between chip rate changes
hw	Maximum allowed deviation of chip rate from average chip rate
write_jitter	1 to write instantaneous chip rate and chip release impulses to files, 0 otherwise
fda	Data rate
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

walib/j_quatary_ds

LEADING EDGE MOD N IMPULSE TRAIN

DESCRIPTION

Outputs an impulse train with output values equal to zero, except for one sample each period when the output is one. Differs from the SPW Mod N Impulse Train in that the output on the first iteration, sample number zero, is one.

MENU PATH

Utilities > Source > Lead Mod N Imp Train

INPUT PORT

None

OUTPUT PORT

OUT	Real

PARAMETERS

period	Period of the impulse train measured in samples. If the parameter value is not an integer the period used is the nearest integer.

LIBRARY / FUNCTION

utlib/lead_train

LOWPASS/BANDPASS FILTER

DESCRIPTION

Functions as either a lowpass or a bandpass filter, with the user making the choice by setting an integer flag to 0 or 1 respectively.

MENU PATH

Utilities > Signal Processing > Low/Bandp Filter

INPUT PORT

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

control	0 for lowpass or 1 for bandpass
lp_edge	Lowpass edge frequency
bp_center	Bandpass center frequency
bp_bw	Bandpass bandwidth frequency
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

utlib/lp_bp_filter

LOWPASS CYCLIC FEATURE DETECTOR SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form, a signal present input to one detector and a signal absent input to another detector. The detectors are each cyclic feature detectors for lowpass cyclic frequencies. The result of processing the signal present case is the H_1 output and the result of processing the signal absent case is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Cyclic Feature > Lowpass, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H2	Complex

PARAMETERS

cyclic_freq	Cyclic frequency
filter_zone	Spectral frequency zone: 0 for lowpass, 1 for bandpass
lp_edge	Lowpass edge frequency
bp_center	Bandpass center frequency
bp_bw	Bandpass bandwidth frequency
delay_time	Delay in delay, conjugate and multiply section of detector systems
collect_time	Output integration time in detector systems
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise

sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	data rate

LIBRARY / FUNCTION

delib/cyc_feat1

LOWPASS CYCLIC FEATURE DETECTOR CORE

DESCRIPTION

Simulates the action of a cyclic feature detector for lowpass cyclic frequencies envelope input. Provides optional storage of intermediate signals.

MENU PATH

Interceptor > Detector > Cyclic Feature > Lowpass, core

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

cyclic_freq	Cyclic frequency
filter_zone	Spectral frequency zone: 0 for lowpass, 1 for bandpass
lp_edge	Lowpass edge frequency
bp_center	Bandpass center frequency
bp_bw	Bandpass bandwidth frequency
delay_time	Delay in delay, conjugate and multiply section of detector systems
collect_time	Output integration time in detector systems
sfreq	Sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/cyc_feat1_core

LOWPASS CYCLIC SPECTRUM

DESCRIPTION

Computes the lowpass component of the cyclic spectrum of a complex signal SDE. The lowpass cyclic spectrum is computed at a user-specified cyclic frequency and for spectral frequencies between $f_0 - f_{sa}/2$ and $f_0 + f_{sa}/2$ where f_{sa} is the sampling frequency and f_0 is the signal center frequency.

The estimation of cyclic spectra combines the analysis of Appendix E with functions in COMDISCO's Signal Display Editor, SDE. The final implementation in SDE is a macro which has performed successfully in test, although errors in COMDISCO's documentation hinder an analytic proof of correctness.

MENU PATH

SDE: SYSTEM/MACRO/EXECUTE/vmlib/Cyclic_lowpass

INPUT PORTS

OUT	Real

OUTPUT PORTS

H1	Complex
H2	Complex

PARAMETERS

numsigs	Display number of signal to analyze
alpha	Cyclic frequency desired
samp_freq	Sampling frequency
numpts	Number of samples to use in the computation
numffpts	Number of points in the FFT (must be a power of two)

LIBRARY / FUNCTION

spwdata/sdemacro/vmilb/Cyclic_low.mac

LOWPASS (KAISER) FILTER

MENU PATH

Spread Spectrum > Lowpass (Kaiser) Filter

INPUT PORTS

IN	Real
DELAY_IN	Real

OUTPUT PORTS

OUT	Complex
DELAY_OUT	Real

PARAMETERS

f_c	Cutoff frequency
delta_f	Transition region width
a_r	Stopband attenuation
sfreq	Sampling frequency

LIBRARY / FUNCTION

walib/kaiser_filter

MAGNITUDE SQUARED

DESCRIPTION

Outputs the magnitude squared of the complex input.

MENU PATH

Utilities > Signal Processing > Magnitude Sqaured

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

OUT	Real

PARAMETERS

None

LIBRARY / FUNCTION

utlib/mag_sqd

MAKE ROC

DESCRIPTION

Treats each pair of inputs as the outputs of the exceedance module, that is as a vector of exceedances and an underflow, and uses each pair of inputs to estimate the probability distribution function of the corresponding real siganl. There are two pairs of inputs, one for the signal present and the other for the signal absent simulation, and the module combines the two probability distribution estimates to form probability of false alarm, probability of detection, pairs. The user specifies a name, **`filename`**, in the parameter table, and the module stores probability paris in that file, in the directory **`/pspwdata/vmsigs`**. To control statistical signifigance, the user specifies a minimum exeedance count number for the signal absent input, and the module terminates the tail of the ROC at the bin where the number of counts goes below the minimum. In addition, the module writes the complete set of probability pairs to a second file with a name formed by adding the extension **`.all`** to **`filename.`**

MENU PATH

None

INPUT PORTS

H1_UNDER	Real
H1_BINS	Real vector
H0_UNDER	Real
H0-BINS	Real vector, of dimension equal to dimension of H1_BINS
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

DEFAULT_VECLEN

Number of components in the input vectors, equal to the number of probability pairs stored in the file **`filename.all.`**

H0_threshold Minimum number of signal absent exceedance counts for bins to be included in the file `filename`

filename Name of file where the module stores probability pairs calculated from bins with signal absent counts exceeding the minimum.

LIBRARY / FUNCTION

vulib/store_roc

Coded in C

MMSE WHITENING FILTER

DESCRIPTION

Implements an adaptive whitening matched filter, which acts to excise interference by estimating the signal plus noise power spectral density and inverting it in the frequency domain. Rather than actually excising interferers, this module deemphasizes the frequency bands in which they fall. Blocking effects are mitigated by overlapping successive blocks to which a complementary taper has been applied.

MENU PATH

Communicator > Preprocessor > MMSE

INPUT PORTS

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

blocksize	Length of FFT
average_type	Periodogram averaging type, either 'sliding_block' or 'decaying'
averaging_block	Length of sliding window of 'sliding_block' averaging type
danielle_param	Width of frequency domain smoothing window
alpha	Decay parameter of 'decaying' average type
window_type	Tapering window applied to lapped reconstruction blocks, either 'hanning' or 'bartlett' are recommended
sfreq	Sampling frequency

LIBRARY / FUNCTION

prlib/lapped_adapt

MODE 1

DESCRIPTION

Estimates the full output SNR, defined in Appendix F as

$$(S/N)_0 = \frac{\left| E\{z \mid H_1\} - E\{z \mid H_0\} \right|^2}{Var\{z \mid H_1\}}$$

and writes the final estimate to a file. The user specifies the filename as a parameter.

MENU PATH

Interceptor > Metrics > Mode 1

INPUT PORTS

H1	Complex
H0	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

filename	Filename to store results

LIBRARY / FUNCTION

vulib/mode_1

MODE 2

DESCRIPTION

Estimates the full output SNR, per collected bit, and writes the result to a file. The full output SNR is defined as

$$(S/N)_0 = \frac{\left| E\{z\,|H_1\} - E\{z\,|H_0\} \right|^2}{Var\{z\;|H_1\}}$$

and the number of collected symbols is the product of the data rate and the collect time.

The Vulnerability Metric extrapolates the result to a range of collect times. After the simulation, a window appears prompting the user to obtain a Gnuplot screen plot by entering

```
Plot2  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

```
Plot2  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`Plot2` is a Bourne shell script which selects and edits the file **`Plot2.hard.template`** if the flag **`-h`** is present, and **`Plot2.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`Plot2.gnu`**. The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Plot2`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Metrics > Mode 2

INPUT PORTS

H1	Complex

MODE 2, continued

H0	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

filename	Filename to store results
sfreq	Sampling frequency
fda	Data rate
collect_time	Collect time
collects	Number of collects

LIBRARY / FUNCTION

vulib>mode_2

MODE 3

DESCRIPTION

Estimates the full output signal noise ratio defined as

$$\frac{\left|E\{z\,|Sig\}\right|^2}{Var\{z\;|H_0\}}$$

and stores the result in a file as a coefficient which permits extrapolation of the result to a range of collect times and a range of interceptor's input signal power. The extrapolation to a range of signal powers is accurate only for low signal powers, and the module ensures that results are presented over a range where they are accurate.

`Plot3 filename`

in an operating system window, where **`filename`** is the data file name selected by the user. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

`Plot 3 -h filename`

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`Plot3` is a Bourne shell script which selects and edits the file **`Plot3.hard.template`** if the flag **`-h`** is present, and **`Plot3.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`Plot2.gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Plot3`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Metrics > Mode 3

INPUT PORTS

SIG	Complex
H0	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

filename	Filename to store results
store_signals	Store signals?
parent	Parent module
fda	Data rate
collect_time	Collect time
sfreq	Sampling frequency
collects	Number of collects

LIBRARY / FUNCTION

vulib>mode_3

MODE 4

DESCRIPTION

Estimates the output signal noise ratio defined as

$$\frac{\left|E\{z\,|Sig\}\right|^2}{Var\{z\ |H_0\}}$$

and stores the result in a file as a coefficient which permits extrapolation of the result to a range of collect times and a range of interceptor's input signal power. The extrapolation to a range of signal powers is accurate only for low signal powers, and the module ensures that results are presented over a range where they are accurate.

The module uses the results of analytic calculations to determine the denominator, thereby shortening the computation in two ways. First the signal absent case does not need to be simulated, and second, estimates of the mean converge faster than estimates of the variance, so the simulation does not have to be as long to achieve a given accuracy.

The extrapolation to a range of signal powers is accurate only for low signal powers, and the module ensures that results are presented over a range where they are accurate.

The user also has the option of storing intermediate statistics on disk. The user selects the filenames.

After the simulation, a window appears prompting the user to obtain a Gnuplot screen plot by entering

```
Plot4  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

```
Plot4  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close. **`Plot3`** is a Bourne shell script which selects and edits the file **`Plot3.hard.template`** if the flag **`-h`** is present, and **`Plot3.soft.template`** otherwise, and calls Gnuplot to execute the

MODE 4, continued

resulting Gnuplot script, `Plot2.gnu`. The data file and the script files are in the directory `/spwdata/vmsign`, but `Plot3` can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Metrics > Mode 4

INPUT PORTS

SIG	Complex
NOISE_BW	Real
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

com_ran_pow	Communicator's range input power. Fixed at 1.0 in SSVM
filename	Filename to store results
fda	Data rate
com_dB_in	Communicator's input SNR, dB
int_ran_adv	Interceptor's noise advantage, dB of power
int_nse_adv	Interceptor's noise advantage, dB of power
collect_time	Collect time
collects	Number of collects
sfreq	Sampling frequency

LIBRARY / FUNCTION

vulib>mode_4

MODE SELECTION

DESCRIPTION

Logically processes the parameter values, and selects one of four outputs to set to low, and sets the rest to high.

MENU PATH

Interceptor > Metrics > Mode Selection

INPUT PORT

None

OUTPUT PORTS

HOLD_1	Real
HOLD_2	Real
HOLD_3	Real
HOLD_4	Real

PARAMETERS

long_stat_coll	Long stationary collect?
quad_weak	Quadratic processing, with weak signal?
ideal_bkgd	Ideal background?

LIBRARY / FUNCTION

vulib > select_mode

MSK DEMODULATOR

DESCRIPTION

Demodulates an MDSK waveform, producing a binary sequence.

MENU PATH

Communicator > Demodulator > MSK

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Real

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/msk_demod

MSK MODULATOR

DESCRIPTION

A minimum shift keyed modulator, implemented as shaped offset QPSK.

MENU PATH

Communicator > Modulator > MSK

INPUT PORT

IN	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/msk_mod

MSK SOURCE

DESCRIPTION

Outputs a narrowband complex envelope resulting from interleaving a pseudorandom bit source, and using the result as the data stream for a binary phase shift keying modulation. The user sets the numbers of columns and rows to define the interleaving, and accepting the default value of 1 for both produces a non-interleaved data stream.

MENU PATH

Communicator > Modulated Source > MSK

INPUT PORT

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

ncols	Interleaver columns
nrows	Interleaver rows
fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modulib/msk_source

NYQUIST FILTERED WAVEFORM

DESCRIPTION

Implements realizable finite impulse response low pass filtering by convolving its input with a truncated sin$(x)/x$

MENU PATH

Utilities > Nyquist Filter

INPUT PORT

IN	Real

OUTPUT PORT

OUT	Real

PARAMETERS

bw	Bandwidth of Nyquist filter
impulse_length	Length in seconds of impulse response
sfreq	Sampling rate

LIBRARY / FUNCTION

utlib/nyq_filter

Coded in C

COMPOSITE OUTPUT SNR METRICS

DESCRIPTION

Logically processes the parameter values describing the interception scenario, and selects one of four output SNR modules, MODE 1, MODE 2, MODE 3, or MODE 4, to process the incoming signals.

MENU PATH

Interceptor > Metrics > Composite Output SNR

INPUT PORTS

H1	Complex
H0	Complex

OUTPUT PORTS

None

PARAMETERS

filename	Filename to store results
com_ran_pow	Communicator's range input power. Fixed at 1.0 in SSVM
int_ran_adv	Interceptor's noise advantage, dB of power
long_stat_coll	Long stationary collect?
quad _weak	Quadratic processing with weak signal?
ideal_bkgd	Ideal background
collect_time	Collect time
collects	Number of collects
sfreq	Sampling frequency
fda	Data rate
com_dB_in	Communicator's input SNR, dB

USAGE SNR METRICS

SSVM>Demonstrations>Correlation Radiometer
SSVM>Demonstrations>Total Power Radiometer

LIBRARY / FUNCTION

vulib>snro_metrics

OVERFLOW PROTECT

DESCRIPTION

Replaces the input with 1 during an initial transient of length specified by the user. After the transient the output equals the input. Avoids transient overflow errors for example if the input is a denominator and equal to zero at the beginning of the simulation.

MENU PATH

Utilities > Logic > Overflow Protect

INPUT PORTS

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

init_protd_tm	Length if transient
sfreq	Sampling rate

LIBRARY / FUNCTION

utlib/ovrflw_protect

PHASOR

DESCRIPTION

Outputs a phasor z_n with frequency and phase exactly as given by the respective inputs f_n and φ_n

$$z_n = e^{j(2\pi f_n n / f_{sa} + \phi_n)}$$

where f_{sa} is the simulation sampling frequency. The frequency is not adjusted to be the inverse of an integer number of simulation sampling intervals.

MENU PATH

Utilities > Sources > Phasor

INPUT PORT

FREQ	Real
PHASE	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/phasor

POWER COMPENSATION

DESCRIPTION

Adaptively multiplies the signal by a scaler to maintain a desired power. The user sets the desired power as a parameter, and the module estimates the incoming power. The user can also specify a start time to permit transients to decay before power compensation is performed.

MENU PATH

Spread Spectrum > Power Compensation

INPUT PORT

IN	Complex
DELAY_IN	Real

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

time_on	Time on
ave_pow	Desired average power
sfreq	Sampling frequency

LIBRARY / FUNCTION

walib/power_compense

QUARTERNARY DS SPREADER

DESCRIPTION

Multiplies the complex input by a quaternary complex pseudonoise sequence specified by the user as a multiple of the data rate. If the input signal has constant unit magnitude, the output power is 1.

Delays the output signal by the input delay value for automatic synchronization.

Outputs the spreading sequence as a global output connector called 'code'.

MENU PATH

Spread Spectrum > Direct Sequence > Quaternary

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

nch	Number of chips per symbol
fda	Data rate
sfreq	Simulation sampling frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

walib/quatary_ds

QPSK DEMODULATOR

DESCRIPTION

Demodulates an quaternary PSK waveform, producing a binary sequence.

MENU PATH

Communicator > Demodulator > QPSK

INPUT PORT

IN	Complex

OUTPUT PORT

OUT	Real

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/qpsk_demod

QPSK MODULATOR

DESCRIPTION

A quaternary PSK modulator.

MENU PATH

Communicator > Modulator > QPSK

INPUT PORT

IN	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modlib/qpsk_mod

QPSK SOURCE

DESCRIPTION

Outputs a narrowband complex envelope resulting from interleaving a pseudorandom bit source, and using the result as the data stream for a binary phase shift keying modulation. The user sets the numbers of columns and rows to define the interleaving, and accepting the default value of 1 for both produces a non-interleaved data stream.

MENU PATH

Communicator > Modulated Source > QPSK

INPUT PORT

None

OUTPUT PORT

OUT	Complex
DELAY_OUT	Real

PARAMETERS

ncols	Interleaver columns
nrows	Interleaver rows
fda	Bit rate
sfreq	Sampling rate

LIBRARY / FUNCTION

modulib/qpsk_source

RANDOM DIGITS

DESCRIPTION

Produces a pseudorandom sequence of nonbinary numbers, using a linear congruential generator. Output range may be as large as 2^{24}-1.

MENU PATH

Utilities > Source > Random Digits

INPUT PORT

None

OUTPUT PORT

OUT	Real

PARAMETERS

symbol_size	Range of possible outputs. One more than the maximum possible value.
starting_state	Initial value.

LIBRARY / FUNCTION

utlib/random_digits

Coded in C

RANDOM INTERFERENCE

DESCRIPTION

Randomly generates an interference configuration and adds the resulting interference to the signal. The user specifies the number of interferers, and the maxima and minima of the interferer width and carrier frequency distributions, and the mean and standard deviation of the interferer power distribution. A configuration of interferers is then randomly generated from the specified distributions and the resulting interference background is added to the signal. The frequency and bandwidth of each interferer are generated from uniform distributions with upper and lower limits specified by the user. The power of the interferers is generated from a power law probability density, with the mean power and the variance specified by the user. The bandwidth is implemented by BPSK modulating each interference carrier at a modulation frequency equal to the bandwidth. The interference configuration is stored in a file. The user may specify the filename, which defaults to rand.conf..

The interference is given by

$$\sum_{=1}^{N} \sqrt{2P_i}\, e^{j2\pi(f_i - f_0)n / f_{sa}}\, b_i\left(Floor\left(\frac{nW_i}{f_{sa}}\right)\right)$$

where n is the sample number, N is the number of interferers, f_{sa} is the simulation sampling frequency, f_0 is the center frequency of the communication signal, and P_i, f_i, W_i and b_i are respectively the power, carrier frequency, one-sided bandwidth and a pseudonoise binary data sequence of the i th interferer. The data sequence has values ± 1.

MENU PATH

Background > Interference > Random

INPUT PORT

None

OUTPUT PORT

OUT Complex

OUTPUT FILE

data/rand.conf

PARAMETERS

if_num Interferers

if_cent_freq Configuration center frequency

if_bandwidth Configuration one-sided bandwidth

if_res Frequency resolution. Carrier frequencies are selected at random from a gridwidth spacing equal to the frequency resolution

if_min Minimum interferer width

if_max Maximum interferer width

if_mean Mean interferer power, in dB signal. If the mean interferer power is a factor of is E above the signal power, if_mean is equal to $10\log_{10}(E)$.

if_std The standard deviation, in dB, of the distribution of interferer power. If the standard deviation of the power distribution is σ, if_std is equal to $10\log_{10}(\sigma)$.

simtime Simulation time

sfreq Sampling rate

fces Signal center frequency

LIBRARY / FUNCTION

bklib/ifrand

Coded in C

READ FROM FILE

DESCRIPTION

Successfully outputs the numbers form a column of real numbers in file named by the user. If the hold port is high the progression through the file stops until the hold goes low again.

MENU PATH

Utilities > Input/Output > Read From File

INPUT PORTS

HOLD	Real

OUTPUT PORT

OUT	Real

PARAMETERS

filename	Name of file containing column of numbers

LIBRARY / FUNCTION

utlib/read_from_file

Coded in C

REAL INPUT SWITCH

DESCRIPTION

Selects the output from one of the real inputs. Parameters specify both the number of permitted inputs, and which line is used for the output. The module tests the value of the both parameters, and terminates with an error message if either is out of range. Setting the parent module parameter permits the error message to give the containing module of the instance that has an error.

Coded in C.

MENU PATH

Utilities > Logic > Switch, Real, Parameter

INPUT PORT

IN_0	Real
IN_1	Real
IN_2	Real
IN_3	Real
IN_4	Real
IN_5	Real

OUTPUT PORT

OUT	Real

PARAMETERS

no_of_inputs	Number of inputs (1-6)
index	Selection index (0- no_of_inputs-1)

LIBRARY / FUNCTION

utlib/switch_in_real

RECEIVER

DESCRIPTION

Simulates a real receiver or tuner, which filters the incoming real signal to reduce noise contamination of the detection processing, and converts the result to a complex envelope form centered on the center frequency of the input passband. Since the entire simulation uses complex envelope representation, the simulation model of the real bandpass filtering is complex lowpass filtering, The Receiver module, by incorporating the Center Frequency between the receiver center frequency and the signal center frequency. The filter model used is a Butterworth IIR filter with an adjustable order set at 20.

MENU PATH

Utilities > Signal Processing > Receiver

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

bwif	Sampling frequency
fces	Signal center frequency
fced	Receiver center frequency
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/receiver

RICIAN ROC

DESCRIPTION

Samples the output of a detector at intervals equal to the detector's collect time; forms estimates of the mean and variance of the incoming signal; treats the magnitude of the mean as the amplitude of a tone, and the variance as the variance of Gaussian noise; and computes the receiver operating characteristic (ROC) of the detection of the tone in the Gaussian noise using an envelope centered on the tone frequency. The resulting ROC is an approximation to the true ROC because the envelope used may not be exactly centered on the feature frequency, and the detector output noise will in general not be exactly Gaussian. By setting the flag in the parameter table, the user may also store the progressive estimates of the variance and the amplitude of the mean for subsequent viewing in SDE. The user specifies the name, **`filename`** , of the file to which the results are written.

After the simulation, the user enters

```
Rician  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user in the spectrum module. The result is a Gnuplot screen plot showing spectral strength as a function of frequency. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

```
Rician  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

The module writes the mean and variance to files called **`rician_roc_mean.dat`** and **`rician_roc_mean.dat`** respectively. When the simulation is over, the custom coded module STORE RICIAN ROC RESULTS reads both files and writes the variance and the amplitude of the mean to a single file with name, **`filename`**, specified by the user in the parameter table of RICIAN ROC. **`Rician`** is a Bourne shell script which selects and edits the file **`Rician .hard.template`** if the flag **`-h`** is present, and **`Rician .soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script,**`Rician .gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Rician`** can be invoked from any directory

A second Bourne shell script also permits the user to integrate communication performance results with the Rician ROC. After the simulation, the user obtains a screen plot by entering

```
Rician filename1 filename2
```

where the additional filename, **`filename2`**, is the name specified by the user in the BIT ERROR RATE module to store the bit error rate estimate. The functionality of **`BerRic`** is similar to the functionality of **`Rician`** , with the addition that the bit error rate estimates and the name of the file used to store it are both shown in the title of the plot. As with **`Rician`**, using the flag -h produces a hard copy as well as the screen plot.

MENU PATH

Interceptor > Metric > Rician ROC

INPUT PORT

H1	Complex

OUTPUT PORTS

None

PARAMETERS

write_moments	Store progressive moment estimates if equal to 1, do not if equal to 0
filename	Name of file to store the moment estimates
collect_time	Collect time used in the detector module
sfreq	Sampling frequency
nsamples	Number of iterations in the simulation

LIBRARY / FUNCTION

vulib/roc_rician

ROC

DESCRIPTION

Forms histograms of the magnitudes of samples of two input, one for signal present and the other for signal absent, and uses the results to estimate the receiver operating characteristic (ROC). The sampling interval is equal to the collect time used in the detector module. The user specifies the lowest and highest values of the detection threshold to be considered, the number of bins into which the intervening range is divided, and the name, **`filename`**, of the file to which the results are written..

After the simulation, the user enters

`ROC filename`

in an operating system window, where **`filename`** is the data file name selected by the user in the spectrum module. The result is a Gnuplot screen plot showing spectral strength as a function of frequency. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

`ROC -h filename`

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`ROC` is a Bourne shell script which selects and edits the file **`ROC.hard. template`** if the flag **`-h`** is present, and **`ROC.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script, **`ROC.gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`ROC`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Metrics > Rician ROC

INPUT PORT

H1	Complex
H0	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

filename	Name of file to store the moment estimates
no_of_bins	Number of bins dividing region between minimum and maximum threshold
min	Lowest considered threshold
max	Highest considered threshold
collect_time	Collect time used in the detector module
sfreq	Sampling frequency
nsamples	Number of iterations in the simulation

LIBRARY / FUNCTION

vulib/roc

SCANNER

DESCRIPTION

Cyclically scans a set of frequency channels and stops when the average power detected exceeds a threshold. The scanner frequency then remains fixed until the average power drops below the threshold, when the scanning continues. The channel center frequencies are either uniformly spaced in a band specified by the user, or listed in a file supplied by the user.

The file must exist in /caedata/data. The power average is a moving average over an interval equal to the channel dwell time. The threshold is a user-specified multiplier times the average power obtained from a reference signal that is typically known to contain no signal.

The user has the options to store intermediate signals inside the scanning radiometer and inside the reference radiometer. The Vulnerability Metric stores the signals in /spwdata/vmsigs, with filenames based on the stem supplied by the user.

MENU PATH

Interceptor > Detector > Radiometer> Scanner

INPUT PORTS

H1	Complex
RFNCE	Complex
HOLD	Real

OUTPUT PORTS

FREQ	Real

PARAMETERS

filename	Name of file containing channel center frequencies
file_or_auto	1 to scan a band at regular intervals, 0 to read center frequencies from the file
auto_scan_bw	Auto scan bandwidth
scan_cent_freq	Center frequency of the scan

channel_bw	Full bandwidth of each channel
dwell_time	Time spent at each frequency channel
threshold	Number of dB above the noise reference to halt scanning
store_scan_rad	Store scanning radiometer signals? (1-yes, 0-no)
store_ref_rad	Store reference radiometer signals? (1-yes, 0-no)
stem	Filename stem for storing signals (no)
sfreq	Sampling frequency
fces	Signal center frequency

USAGE EXAMPLE

SSVM>Demonstrations>Scanner

LIBRARY / FUNCTION

delib/scanner

SINE WAVE

DESCRIPTION

Outputs a sine wave with frequency and phase determined by the respective input signals. Unlike the SPW Function Generator, the Vulnerability Metric Sine Wave module does not change the frequency to make the period an integer of sampling intervals.

MENU PATH

Utilities > Sources > Sine Wave

INPUT PORTS

FREQ	Real
PHASE	Real

OUTPUT PORT

OUT	Real

PARAMETERS

sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/sine_wave

SPECTRA

DESCRIPTION

Filters and subsamples the input, then forms the magnitude squared of the Fourier Transform of the subsampled signal, and finally performs a sliding block average of the resulting spectrum. The process repeats and the averaged spectra are stored in ASCII format in a data file with name specified by the user as a parameter. The user also specifies the value of a spectrum width parameter and the actual spectrum width is rounded up to a submultiple of the sampling frequency.

$$\text{Actual Spectrum Width} = \frac{\text{Sampling Frequency}}{\text{Floor (Sampling Frequency/ Spectrum Width Parameter)}}$$

The bandwidth of the input filter is set equal to the Actual Spectrum Width to eliminate aliasing, resulting in spectrum edges contaminated by the filter band edges. As an aid to the user, updating an additional parameter shows the duration of signal input required to form each averaged spectrum.

After the simulation, the user enters

```
Waterfall  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user in the spectrum module. The result is a Gnuplot screen plot showing spectral strength as a function of frequency. The plot also displays the name of the underlying data file, To obtain a hard copy in addition to the screen plot, the user enters

```
Waterfall  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`Waterfall` is a Bourne shell script which selects and edits the file **`Waterfall.hard.template`** if the flag **`-h`** is present, and **`Waterfall.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script,**`Waterfall.gnu.`** The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Waterfall`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Detector > Spectral > Spectra

INPUT PORT

IN	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

spectrum_width	Approximate width of final spectra
fft_size	Number of points in spectrum, power if 2
window_tupe	Windowing method used in FFT
spec_avg_block	Size of sliding block used to average spectra
fced	Input filter center frequency
filename	Name of file to store the moment estimates
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

delib/spectra

SPECTRAL ESTIMATE AND MATCH

DESCRIPTION

Custom coded module preforms the frequency domain operations necessary to implement the adaptive whitening filter. It accepts a frequency domain block, updates the running spectral estimate, and divides componentwise by that estimated to form a frequency domain output.

MENU PATH

None

INPUT PORTS

X_re	Real vector
X_im	Real vector
HOLD	Real

OUTPUT PORTS

Y_re	Real
Y_im	Real

PARAMETERS

blocksize	Number of points used in the FFTs
average_type	Periodogram averaging type, either 'sliding_block' or 'decaying'
average_block	Length of sliding window for 'sliding_block' averaging type
daniell_param	Width of frequency domain smoothing window
nexempt	Number of points excluded in the calculation of mean square magnitude
alpha	Decay parameter for 'decaying' averaging type
sfreq	Simulation sampling frequency

LIBRARY / FUNCTION

prlib/spectral_est

CODED IN C

SPECTRUM

DESCRIPTION

Starts at a time specified by the user, filters and subsamples the input, then forms the magnitude squared of the Fourier Transform of the subsampled signal, and finally performs a sliding block average of the resulting spectrum. The averaged spectra are stored in ASCII format in a data file with name specified by the user as a parameter. The user also specifies the value of a spectrum width parameter and the actual spectrum width is rounded up to a submultiple of the sampling frequency.

$$\text{Actual Spectrum Width} = \frac{\text{Sampling Frequency}}{\text{Floor (Sampling Frequency/ Spectrum Width Parameter)}}$$

The bandwidth of the input filter is set equal to the Actual Spectrum Width to eliminate aliasing, resulting in spectrum edges contaminated by the filter band edges. As an aid to the user, updating an additional parameter shows the duration of signal input required to form each averaged spectrum.

After the simulation, the user enters

```
Spectrum  filename
```

in an operating system window, where **`filename`** is the data file name selected by the user in the spectrum module. The result is a Gnuplot screen plot showing spectral strength as a function of frequency. The plot also displays the name of the underlying data file. To obtain a hard copy in addition to the screen plot, the user enters

```
Spectrum  -h  filename
```

If the default printer has not been set during the session, the user may have to specify the printer to be used by entering, in the operating system window, **`setenv PRINTER`** followed by the printer name. Quitting the graphics window, or typing a keystroke in the operating system window causes the graphical output to disappear, and the Gnuplot session to close.

`Spectrum` is a Bourne shell script which selects and edits the file **`Spectrum.hard.template`** if the flag **`-h`** is present, and **`Spectrum.soft.template`** otherwise, and calls Gnuplot to execute the resulting Gnuplot script,**`Spectrum.gnu`**. The data file and the script files are in the directory **`/spwdata/vmsign`**, but **`Spectrum`** can be invoked from any directory without specifying a path.

MENU PATH

Interceptor > Detector > Spectral > Spectra

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

None

PARAMETERS

start_time	Time to begin collecting data to determine averaged spectrum
spectrum_width	Approximate width of final spectra
fft_size	Number of points in spectrum, power if 2
window_tupe	Windowing method used in FFT
spec_avg_block	Size of sliding block used to average spectra
fced	Input filter center frequency
filename	Name of file to store the moment estimates
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

delib/spectrum

SQUARE WAVE

DESCRIPTION

Outputs a square wave z_n with frequency exactly as given by the frequency input f_n

$$z_n = \text{SquareWave}\,(2\pi f_n\ n / f_{sa})$$

where f_{sa} is the simulation sampling frequency. In contrast to the SPW implementation of function generators, the frequency is not rounded to the inverse of an integer number of simulation sampling intervals.

MENU PATH

Utilities > Sources > Square Wave

INPUT PORT

FREQ	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/square_wave

STEPPER

DESCRIPTION

Produces a stepped output which cycles through a set of values specified by the user. In the auto mode, the values are regularly spaced and specified by the starting value, the increment, and the number of values. Alternatively the user may store arbitrary values in a file in the directory `/caedata/data/`, and specify a scaling factor and a offset to be applied to the values in the file. The user controls the name of the file, and the dwell time, which is the time interval between changes in the output.

A file with the chosen name must be present in `/caedata/data/` , even if it is not used because the user chooses the auto mode.

MENU PATH

Utilities > Sources > Stepper

INPUT PORT

HOLD	Real

OUTPUT PORTS

OUT	Real

PARAMETERS

file_or_auto	1 to scan a band at regular intervals, 0 to read center frequencies from the file
dwell_time	Time spent at each frequency channel
start_freq	Initial output value for auto mode
step_freq	Step in output value for auto mode
no_of_values	Number of values for auto mode
filename	Name of file containing channel center frequencies
scale_factor	Scaling applied to values in file
offset	Offset applied to scaled values from file
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/stepper

STORE COMPLEX VECTORS

DESCRIPTION

Provides optional storage of intermediate complex vectors for subsequant analyisis

MENU PATH

Utilities > Input/Output > Store Complex Vectors

INPUT PORTS

IN1	Complex
IN2	Complex
IN3	Complex
HOLD	Real

OUTPUT PORT

None

PARAMETERS

blocksize	Vector length
no_of_vectors	Number of vectors to store
filename_stem	Filename stem
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/sto_cmplx_vecs

STORE DETECTOR DATA

DESCRIPTION

Stores detector parameter settings to files on disk.

MENU PATH

Interceptor > Detector > Utilities > Store Detector Data

INPUT PORT

None

OUTPUT PORT

None

PARAMETERS

bwif	Sampling frequency
fchd	Interceptor's chip rate estimate
nsamples	Number of samples in simulation
fda	Data rate
collect_time	Output integration time in detector systems

LIBRARY / FUNCTION

delib/store_det_data

STORE ON TERMINATING

DESCRIPTION

Writes the signal value at the IN port at the end of the simulation to a file. The user specifies the name of the file as a parameter, and the file is written to the Graphics Directory. The Graphics Data directory is specified in the file /spwsys/vminclude/all.h, and is the same for all graphics output data.

MENU PATH

Interceptor > Metrics > Store on Terminating

INPUT PORTS

IN	Real
HOLD	Real

OUTPUT PORT

None

PARAMETERS

filename	Filename

LIBRARY / FUNCTION

vulib/sto_on_termng

STORE POWER

DESCRIPTION

Computes the power of the input signal, averaged over the whole simulation result in a file on disk.

MENU PATH

Utilities > Input/Output > Store Power

INPUT PORTS

IN	Complex

OUTPUT PORT

None

PARAMETERS

nsamples	Number of iterations in simulation
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/store_power

STORE REAL SIGNALS

DESCRIPTION

Provides optional storage of intermediate real signals for subsequent analysis

MENU PATH

Utilities > Input/Output > Store Real Signals

INPUT PORTS

IN1	Real
IN2	Real
IN3	Real
HOLD	Real

OUTPUT PORT

None

PARAMETERS

no_of_vecs	Number of signals to store (3, or less)
stem	Filename stem
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/store_real

STORE RICIAN ROC

DESCRIPTION

After the simulation completes, writes the final signal value at each of the two inputs to a file, with the value at the ABSMEAN port in the first line of the file, and the value at the VAR port on the second line. The user specifies the name of the file as a parameter, and the file is written to the Graphics Directory. The Graphics Data directory is specified in the file /spwsys/ vminclude/all.h, and is the same for all graphics output data.

Coded in C

MENU PATH

Interceptor > Metrics > Store Rician ROC

INPUT PORTS

ABSMEAN	Real
VAR	Real
HOLD	

OUTPUT PORT

None

PARAMETERS

filename	Name of file to store amplitude of mean and variance

LIBRARY / FUNCTION

vulib/sto_ric_roc

STORE (COMPLEX) SIGNALS

DESCRIPTION

Provides optional storage of intermediate (complex) signals for subsequent analysis

MENU PATH

Utilities > Input/Output > Store Signals

INPUT PORTS

IN1	Complex
IN2	Complex
IN3	Complex
HOLD	Real

OUTPUT PORT

None

PARAMETERS

no_of_vecs	Number of signals to store (3, or less)
stem	Filename stem
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/store

STORE SPECTRA

DESCRIPTION

Stores the input vectors and a vector of frequencies in columns in an ASCII file `/spwdata/vmsigs/`. At each iteration the elements of the input vector at SPECTRA are reordered to place the zero frequency component in the center to the first column in the file. Parameters define the width of the spectrum and frequencies, and the module computes a vector of frequencies and appends the column in the file. The user may zoom in on the input spectrum by only plot the spectrum, centered at zero frequency, with the ratio of the total spectrum, size being a power of two specified by the user. Coded in C.

MENU PATH

Interceptor > Detector > Utilities > Store Spectra

INPUT PORTS

SPECTRA	Real Vector
TIMES	Real Vector
HOLD	Real

OUTPUT PORT

None

PARAMETERS

spectrum_width	Width of spectra
freq_zoom	Zoom factor, power of 2
DEFAULT_VECLEN	
	Number of points in spectrum, power of 2
fced	Input filter center frequency
filename	Name of file to store spectrum

LIBRARY / FUNCTION

delib/store_spectra

STORE THREE DIMENSIONAL DATA

DESCRIPTION

Treats each component of the input vector as the value of a function of two variables. The value of the first variable increments from one iteration to the next, while the value of the second variable increments from one component of the input vector to the next. The module stores the data in a file as triplets, with one triplet per line, and each triplet consisting of the values of the first variable, the second variable and the vector component. Tabs separate numbers on a line. Coded in C.

MENU PATH

Utilities > Input/Output > Store 3d Data

INPUT PORTS

IN	Real vector
HOLD	Real

OUTPUT PORT

None

PARAMETERS

start_1	First variable starting value
step_1	First variable increment value
start_2	Second variable starting value
step_2	Second variable increment value
DEFAULT_VECLEN	Number of different values assumed by the second variable, equals dimension of input vector
filename	Name of file to store spectrum

LIBRARY / FUNCTION

utlib/store_3d_data

SWITCHED RADIOMETER SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form, a signal present input to one detector and a signal absent input to another detector. The detectors are each radiometer detectors, and an additional input port permits the user to supply the switching frequency which both adds to the incoming signal and serves as the switched reference source processing the signal present case is the H_1 output and the result of processing the signal absent case is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Radiometer > Switched, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex
RECVR_NOISE	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver
sw_freq	Switching frequency
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency

SWITCHED RADIOMETER SYSTEM, continued

sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	data rate
collect_time	Output integration time in detector systems

LIBRARY / FUNCTION

delib/sw_radiom

SWITCHED RADIOMETER CORE

DESCRIPTION

Stimulates the action of a switched radiometer detector,with complex envelope additional input permits the user to supply receiver noise, which both adds signal and serves as the switched reference.

MENU PATH

Interceptor > Detector > Radiometer > Switched, core

INPUT PORTS

IN	Complex
RECVR_NOISE	Complex

OUTPUT PORT

OUT	Complex

PARAMETERS

write_signals	1 to store intermediate signals, 0 otherwise
fced	Receiver center frequency
sw_freq	Switching frequency
bwif	Receiver
collect_time	Output integration time in detector systems
sfreq	Sampling frequency
fces	Signal center frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/sw_radiom_core

TAPPED DELAY LINE

DESCRIPTION

Outputs the weighted sum of the input signal and N -1 delayed samples of the input signal. The number of taps, N, equals the integer truncation of the product of the spread T_{sp} and the simulation sampling frequency f_{sa} The weights are read from an input file built by the user. The number of taps must be less than or equal to 512. The output signal $y(n)$ is given by

$$y(n) = \sum_{k=0}^{N-1} c_k x(n-k)$$

where $x(n)$ is the input signal, n is the sample number, c_k are the weights input by the user, and

$$N = \text{Floor}\,(T_{sp} f_{sa})$$

MENU PATH

Channel > Tapped Delay

INPUT PORT

IN	Complex

OUTPUT PORTS

OUT	Complex

INPUT FILE

data/weights_512

Each tap weight is listed on a separate line in the format magnitude phase

PARAMETERS

spread	Largest delay time, in seconds
sfreq	Sampling frequency

LIBRARY / FUNCTION

chlib/tapdelay

Coded in C

TOWER POWER RADIOMETER SYSTEM

DESCRIPTION

Uses a communication signal input and a background input to form, a signal present input to one detector and a signal absent input to another detector. The detectors are each total power radiometer detectors. The result of processing the signal absent case is the H_0 output. Stores detector parameter values in files.

MENU PATH

Interceptor > Detector > Radiometer > Total Power, system

INPUT PORTS

SIGNAL	Complex
BACKGROUND	Complex

OUTPUT PORTS

H1	Complex
H0	Complex

PARAMETERS

fced	Receiver center frequency
bwif	Receiver
collect_time	Output integration time indetector systems
wswgn	1 for weak signal, white Gaussian noise mode, 0 otherwise
nwssb	1 for nonweak signal, simulated background mode, 0 otherwise
fces	Signal center frequency
sfreq	Sampling frequency
nsamples	Number of samples in simulation
fda	data rate

LIBRARY / FUNCTION

delib/radiometer

TOTAL POWER RADIOMETER CORE

DESCRIPTION

Simulates the action of a radiometer detector with complex envelope input. Provides optional storage of intermediate signals.

MENU PATH

Detector > Radiometer > Total Power, core

INPUT PORTS

IN	Complex
HOLD	Real

OUTPUT PORTS

OUT	Complex

PARAMETERS

write_signals	1 to store intermediate signals, 0 otherwise
fced	Receiver center frequency
bwif	Receiver
collect_time	Output integration time indetector systems
sfreq	Sampling frequency
fces	Signal center frequency
nsamples	Number of samples in simulation

LIBRARY / FUNCTION

delib/radiometer_cor

TRIGGERED FREQUENCY HOPPING

DESCRIPTION

Performs incoherent frequency hopping, shifting the incoming signal's frequency and phase by random offsets which change each time the trigger input goes high. If the input signal has constant unit magnitude, the output power is 1.

MENU PATH

Spread Spectrum > Trig Freq Hopping

INPUT PORTS

IN	Complex
TRIGGER	Real
HOLD	Real

OUTPUT PORT

OUT	Complex

PARAMETERS

nfreq	Number of available frequency offsets
maxfreqs	Maximum frequency deviation
write_freqs	1 to store frequency offsets
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

walib/trig_inc_fh

TRUNCATED NYQUIST FILTER

DESCRIPTION

Coded in C.

MENU PATH

Utilities > Trunc Nyquist Filter

INPUT PORT

IN	Real

OUTPUT PORT

OUT	Real

PARAMETERS

f_c	Cutoff frequency
delta_f	Width of transition region
a_r	Stopband attenuation
sfreq	Sampling frequency

LIBRARY / FUNCTION

utlib/kaiser_lpf

TWO DIMENSIONAL STEPPER

DESCRIPTION

Outputs two incrementing reals, with the second output cycling through its entire range each time the first output changes once. The first output does not cycle, but keeps incrementing indefinitely.

MENU PATH

Utilities > Sources > 2D Stepper

INPUT PORT

None

OUTPUT PORTS

OUT1	Real
OUT2	Real

PARAMETERS

start_1	First value output at OUT1
step_1	Increments of the OUT1 value
start_2	First value output at OUT2
step_2	Increments of the OUT2 value
no_of_values_2	Number of distinct values taken at OUT2

LIBRARY / FUNCTION

utlib/two_d_stepper

Coded in C

VECTOR LOG (BASE 10)

DESCRIPTION

Outputs a vector z which is the pointwise log (base 10) of the input vectors x.

$$z_i = \log_{10}(x_i)$$

MENU PATH

Utilities > Vector > Vector Log (Base 10)

INPUT PORTS

IN	Real Vector
HOLD	Real

OUTPUT PORT

OUT	Real vector

PARAMETERS

DEFAULT_VECLEN

Length of input and output vectors

LIBRARY / FUNCTION

utlib/vector_log10

VECTOR MAGNITUDE SQUARED

DESCRIPTION

Outputs a vector z which is the pointwise sum of squares of the input vectors x and y.

$$z_i = x_i^2 + y_i^2$$

where the subscript i denotes the vector index.

MENU PATH

Utilities > Vector > Vector Magnitude Sqd

INPUT PORTS

REAL	Real vector
REAL	Real vector
HOLD	Real

OUTPUT PORT

OUT	Real vector

PARAMETERS

vector_length	Length of input and output vectors

LIBRARY / FUNCTION

utlib/vec_mag_sqd

VECTOR MEAN

DESCRIPTION

Outputs a vector z which is the sliding block average of the input vector x. If the length of the sliding block is N and $z_{n,i}$ denotes the i th component of the input vector at the n th simulation iteration, then

$$z_{n,i} = \sum_{m=0}^{N-1} x_{n-m,i}$$

MENU PATH

Utilities > Vector > Vector Mean

INPUT PORTS

REAL	Real vector
IMAG	Real vector
HOLD	Real

OUTPUT PORT

OUT	Real vector

PARAMETERS

blocksize	Number of input vectors used in sliding block average
DEFAULT_VECLEN	Length of input and output vectors

LIBRARY / FUNCTION

utlib/vec_mean

Coded in C

VECTOR SUBSAMPLER

DESCRIPTION

Creates an output vector by subsampling the input vector. The subsampling ratio is the integer truncation of the ratio between the input and output vector lengths. Using vector indices starting at 1, the output vector contains exactly those elements of the input vector with indices that are an integer multiple of the subsampling ratio.

MENU PATH

Utilities > Vector > Vector Subsampler

INPUT PORTS

IN	Real Vector
HOLD	Real

OUTPUT PORT

OUT	Real vector

PARAMETERS

IN_IOVEC_LEN	Output vector length

LIBRARY / FUNCTION

utlib/vec_subsampler

Coded in C

VHF INTERFERENCE

DESCRIPTION

Adds to the signal a weak medium or strong interference background generated from a stored configuration chosen to be characteristic of VHF operating environments. The user selects the center and width of the operating band, and the module generates the interferers which lie in the selected band.

The mathematical form of the interference the same as that detailed for Random Interference.

MENU PATH

Background > Interference > VHF - Weak

Background > Interference > VHF - Medium

Background > Interference > VHF - Strong

INPUT PORTS

None

OUTPUT PORT

OUT	Complex

INPUT FILES

data/wvhf.conf

data/mvhf.conf

data/svhf.conf

PARAMETERS

if_cent_freq	Configuration center frequency
if_bandwidth	Configuration one-sided bandwidth
simtime	Simulation time
sfreq	Sampling frequency
fces	Signal center frequency

LIBRARY / FUNCTION

bklib/ifwvhf

bklib/ifmvhf

bklib/ifsvhf

Coded in C

WHITENING EXCISION CORE

DESCRIPTION

Combines the two input vectors to form a complex vector, and divides each resulting component by its magnitude. The user specifies the length of the vectors. The spectral resolution, or bin size, is the sampling frequency divided by the FFT size. If the input magnitude of any component is smaller than the machine's arithmatic resolution, the magnitude is set to 1, and the phase is arbritrarily set to 0.

MENU PATH

None

INPUT PORTS

X-re	Real Vector
X-im	Real Vector

OUTPUT PORTS

Y-re	Real Vector
Y-im	Real Vector

PARAMETERS

blocksize	Number of points used in the FFTs
window_type	Type of window used in the forward FFT
sfreq	Sampling frequency

LIBRARY / FUNCTION

prlib/exwhite_cmp

Coded in C

WHITE NOISE

DESCRIPTION

Produces a discrete complex envelope model of continuous bandpass white noise. The noise can be pure Gaussian, pure Gamma, or any mixture of the two, as specified by the user. The noise power in both the Gaussian and Gamma Noise modules is parameterized by the communicator's input signal to noise ratio, which is computed for the case where only one noise module is present. Using more than one noise module at a time requires manually determining the input signal to noise ratio.

MENU PATH

Background > Noise > White

INPUT PORT

None

OUTPUT PORTS

OUT	Complex
HOLD	Real

PARAMETERS

gauss_propn	Gaussian proportion of variance
com_ran_pow	Communicator's range input power, fixed at 1
snr_in	Communicator's input signal-to-noise ratio, dB
sfreq	Sampling frequency
fda	Data rate

LIBRARY / FUNCTION

bklib/white_noise

☆U.S. GOVERNMENT PRINTING OFFICE: 1997-509-127-47110

Appendix D
References

[1] Mac A. Cody, The Fast Wavelet Transform Beyond Fourier Transforms, Dr. Dobb's Journal, Miller Freeman Publishing, 600 Harrison St., San Francisco, CA, 94107, April 1992.

[2] Mac A. Cody, The Wavelet Packet Transform Extending the Wavelet Transform, Dr. Dobb's Journal, Miller Freeman Publishing, 600 Harrison St., San Francisco, CA, 94105, April 1994.

[3] Mac A. Cody, A Wavelet Analyzer An Alternative to the FFT-Based Spectrum Analyzer, Dr. Dobb's Journal, Miller Freeman Publishing, 600 Harrison St., San Francisco, CA, 94105, April 1993.

[4] Mac A. Cody, The Fast Wavelet Transform, Dr. Dobb's Journal, Miller Freeman Publishing, 600 Harrison St., San Francisco, CA, 94105, April 1992.

[5] Ronald R. Coifman, Yves Meyer, and Victor Wickerhauser, Wavelet Analysis and Signal Processing, New Haven, Ct., Yale University, 1991, preprint.

[6] Masami Ueda and Suresa Lodha, Wavlets: An Elementary Introduction and Examples, UCSC-CRL 94-97, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, Santa Cruz, CA, 95064, January 1995.

[7] Bjorn Jawerth and Wim Sweldens, An Overview of Wavelet Based Multi-Resolution Analyses, University of South Carolina, Department of Mathematics, Columbia, SC, 29208, Katholieke Universteit Leuven, Department of Computer Science, Celestijnenlaan 200A, B3001 Leuven, Belgium, August 1993.

[8] Ronald A. DeVore and Bradley J. Lucier, Wavelets, In *Alta Numerica* 1, pages 1-56, 1991.

[9] Adrian Vanzyl, Increasing Web Bandwidth Through Image Compression: An Overview of GIF, JPEG and Fractal Compression Techniques, Unit of Medical Informatics, Monash University, 867 Centre Road, East Bentleigh, 3165, 1995.

[10] Wavelet web site at http://www.amara.com/IEEEwave/IEEEwavelet.html

[11] Wavelet web site at http://eewww.eng.ohio-sta...u/~clin/wavecompress.html

[12] Wavelet web site at http://www.summus.com/wavelets.html

[13] JPEGFAQ - The FAQ for JPEG image compression. FTP access from ftp://rtfm.mit.edu/pub/usenet/news-answers/jpeg-faq.

[14] GIF 89a - Graphics Interchange Format, Programming Reference, Version 89a, Compuserve Incorporated, Graphics Technology Department, 5000 Arlington Center Boulevard, Columbus, Ohio, 43220.

[15] M.R. Nelson, LZW Data Compression, Dr. Dobb's Journal, Miller Freeman Publishing, 600 Harrison St., San Francisco, CA, 94105, October 1989.

[16] J. Ziv and A. Lempel, A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, May 1977.

[17] T. Welch, A Technique for High-Performance Data Compression, Computer, IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1314, June 1984.

[18] COMPFAQ - The FAQ for the comp.compression and comp.compression.research groups. FTP access from ftp://rtfm.mit.edu/pub/usenet/news.answers/compression-faq.

[19] JPEG standard description - Digital Compression and Coding of Continuous-Tone Still Images, Part 1: Requirements and Guidelines, Number: ISO/IEC CD 10918-1.

[20] B. Simon, How Lossy Compression Shrinks Image Files, PC Magazine, Ziff-Davis Publishing Company, L.P., One Park Avenue, New York, NY, 10016, July 1993.

[21] Charles K. Chui, An Introduction to Wavelets, Academic Press, Inc. 1992.

[22] Foley, vanDom, Ferner and Hughes, Computer Graphics - Principals and Practice; Second Edition, Addison Wesley.

[23] H.L. Resnikoff and C.S. Burrus, Relationships Between the Fourier Transform and the Wavelet Transform, *Aware Report No. AD900609.*

[24] R. DeVore, B. Jawerth, and B. Lucier, Image Compression Through Wavelet Transform Coding, IEEE Transactions on Information Theory, 38(2):719-746,1992.

[25] R. DeVore, B. Jawerth, and B. Lucier, Data Compression using Wavelets: Error, Smoothness, and Quantization, DCC-91, Data Compression Conference (J.A. Storer and J.H. Reif, eds.), IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 186-195.

[26] O. Rioul and M. Vetterli, Wavelets and Signal Processing, IEEE Signal Processing Magazine 8 (1991), no. 4, 14-38.

[27] R.R. Coifman, Y. Meyer, S. Quake, and M.V. Wickerhauser, Signal Processing and Compression with Wave Pockets in Proceedings of the conference on Wavelets, Marseille, 1989.

[28] Leigh Sneddon, Paul Avila, Mahendra Mallick, Lawrence Ozarow, Steve Thomas, Robert Pinto, Spread Spectrum Vulnerability Metric, TASC, 55 Walkers Brook Drive, Reading, MA 01867.

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.